# Contributions à la théorie des modèles finis et à la complexité descriptive

Yassine Hachaichi

Université Paris VII - Denis Diderot

Thèse de Doctorat – UFR de Mathématiques

Spécialité : Logique et Fondements de l'Informatique

# Contributions à la théorie des modèles finis et à la complexité descriptive

Yassine Hachaïchi

Soutenance le 13-07-2001

## *Jury*

| | | |
|---|---|---|
| Patrick Cegielski | LACL, université de Paris 12 | Directeur de thèse |
| Etienne Grandjean | GREYC, université de Caen | Rapporteur |
| Georg Gottlob | DBAI, université de Vienne | Rapporteur |
| Serge Grigorieff | LIAFA, université de Paris 7 | |
| Denis Richard | LLAIC, université de Clermont 1 | |
| Arnaud Durand | LACL, université de Paris 12 | |
| Sedki Boughattas | Équipe de Logique Mathématique, université de Paris 7 | |

# Remerciements

Je voudrai commencer par remercier Monsieur Patrick Cegielski pour tout ce qu'il m'a apporté scientifiquement et humainement, et surtout pour sa patience pendant toutes les phases de cette thèse.

Je remercie Messieurs Georg Gottlob et Etienne Grandjean pour leurs patience et l'intérêt qu'ils ont porté à ce travail et leurs conseils sur le fond et la forme qui ont abouti à cette version du travail. Je remercie Messieurs Sedki Boughattas et Arnaud Durand pour avoir accepté de faire partie de mon jury de thèse ; Monsieur Denis Richard pour faire partie de mon jury de thèse et son enthousiasme communiquant pendant les cours de DEA ; Monsieur Serge Grigorieff pour avoir accepté, avec sa gentillesse habituelle, de faire partie de mon jury de thèse et pour les années d'enseignement que j'ai effectuées sous sa direction.

Cette thèse n'aura jamais abouti sans le travail de tous les enseignants que j'ai eu depuis la maternelle. Je voudrai citer ceux qui ont vraiment influencé mon parcours : Mr Noureddine Saïdi, mon prof de math au lycée, Mr René Cori, qui m'a donné mon premier cours de logique, et tous mes profs de DEA de logique et fondements de l'informatique à Paris 7.

Je remercie toute ma famille et mes amis en France et en Tunisie pour leur soutien sans lequel ce travail n'aura jamais vu le jour, je ne peu pas citer tous les noms car ce sera trop long et j'en oublirai sûrement.

Merci à tous les membres de l'équipe de logique et surtout Mr Jean-Pierre Ressayre pour tous ses conseils et son soutien, à Mr Sedki Boughattas qui a subi mes élucubrations (scientifiques et autres), et à Eugenio Chinchilla pour sa bonne humeur contagieuse.

Je remercie tous les membres du LACL qui m'ont accueilli parmi eux, notamment Joelle Cohen, Elizabeth Pelz, Hanna Claudel, et tout le personnel de la MIAGE et de la faculté des sciences économiques et de Gestion de Paris 12, et un grand merci à Tristand Crolard et Arnaud Durand pour leurs sympathie.

Enfin, “Last but not least”, je voudrais dédier cette thèse à mon père qui m’a toujours soutenu (même quand je faisais des bêtises de jeunesse). Je ne pourrais jamais énumérer ses qualités (ni ses défauts, dont j’ai hérité) dans cette thèse car aucun langage (formel ou naturel) ne peut les exprimer. Je lui dis simplement merci.

À mon père, Souaï Lahchaïchi

# Table des matières

# Chapitre 1

# Introduction

## 1.1 Survol du sujet

Ce travail a comme ambition de contribuer à la caractérisation logique de certaines classes de langages, c'est-à-dire à la discipline appelée *complexité descriptive.*

Un *langage* $L$ sur un *alphabet* (c'est-à-dire un ensemble fini non vide d'éléments appelés *symboles*) $\Sigma$ peut être défini de différentes manières. Pour n'en citer que les plus célèbres, voici trois méthodes pour définir un langage :

1. $L$ est un sous-ensemble de l'ensemble de tous les *mots* (les suites finies de symboles de $\Sigma$) qui satisfont une certaine propriété. C'est l'analogue du schéma de compréhension de la théorie des ensembles. Par exemple l'ensemble des mots sur $\Sigma = \{a\}$ dont la longueur (le nombre de caractères) est paire forme un langage.

2. $L$ est un sous-ensemble de l'ensemble de tous les mots qui sont reconnus par un certain modèle de calcul (automate fini, automate à pile, machine de Turing, éventuellement avec une borne sur les ressources, etc). Ceci correspond au *point de vue de l'auditeur* qui doit reconnaître les phrases (mots) du langage. Par exemple, l'automate fini déterministe $\mathcal{A} = (\{a\}, \{q_0, q_1\}, q_0, \{q_1\}, \delta)$, avec $\delta(q_0, a) = q_1, \delta(q_1, a) = q_0$ reconnaît le langage $L$ constitué des mots de longueur paire.

3. $L$ est un sous-ensemble de l'ensemble de tous les mots qui sont générés par une grammaire formelle. C'est le *point de vue du locuteur* qui doit respecter des règles précises de grammaire pour formuler ses phrases. Par exemple le

langage généré par :

$$G = (\{a\}, \{S\}, S, S \to aSa|\varepsilon),$$

où $\varepsilon$ est le mot vide, à ne pas confondre avec l'ensemble vide $\emptyset$, est le langage constitué des mots de longueur paire.

En *théorie de la complexité algorithmique*, les classes de la hiérarchie sont construites par des contraintes sur une, ou plusieurs, ressources utilisées par un certain modèle de calcul pour décider si une instance d'un problème est, oui ou non, vraie, ce qui correspond au point de vue 2 cité plus haut. Généralement, ces bornes sur les ressources sont des fonctions croissantes en la taille de l'entrée, qui dominent les ressources utilisées pour une entrée de cette taille. Les modèles de calcul les plus fréquemment utilisés sont : les automates (finis, à pile, $\cdots$), les machines de Turing, les machines à accès aléatoire et les circuits booléens. Pour plus de détails voir par exemple [Pap94].

*La complexité descriptive* prend une approche diffèrente pour classifier les problèmes. Au lieu de mesurer les ressources nécessaires pour résoudre le problème, cette discipline suggère de mesurer les ressources nécessaires pour le décrire, ceci utilise le point de vue 1 cité plus haut. Le formalisme naturel qui s'impose pour décrire les problèmes est le formalisme logique, au vu de son lien historique à l'informatique et l'existence de hiérarchies syntaxiques et sémantiques en logique. On peut mesurer la complexité d'un problème en fonction du nombre de variables, des types des variables, du nombre de quantificateurs, du type des quantificateurs, etc.

Un des buts de la complexité descriptive est de donner des définitions indépendantes du modèle de calcul des classes de complexité structurelles. Ceci nous assure qu'on a défini des classes robustes et légitimes. On essaye ainsi de généraliser la thèse de Church pour la notion de calculabilité. Une autre direction qui s'impose est de traduire les problèmes de hiérarchie de complexité structurelle en questions sur le pouvoir d'expression de logiques. Il a été prouvé que les plus célèbres classes de complexité structurelles correspondent à des classes de complexité descriptives.

En étudiant le lien entre la complexité en temps sur les machines de Turing et les spectres finis, Fagin [Fag74] a prouvé que la classe $NP$ (celle des problèmes reconnaissables en temps polynomial sur une machine de Turing non-déterministe) correspond exactement aux problèmes définissables par des énoncés de la logique existentielle du second ordre.

À propos du théorème de Fagin, Immerman dit :

> *"This was a remarkable insight, for it* [the result of Fagin] *demonstrated that the complexity of a problem can be understood as the richness of a language needed to specify the problem. Time and space are not model-dependent engineering concepts, they are more fundamental."* ([Imm99, page 1])

## 1.2 Contenu de la thèse

Le prochain chapitre est entièrement consacré aux définition, notations et rappels.

Dans le chapitre 3, on donne une caractérisation logique de la classe des langages algèbriques non-ambigus. On utilisera pour cela une caractérisation logique des langages algébriques due à Lautemann, Schwentick et Thérien [LST95].

Dans le chapitre 4, on prouve, par une comparaison de pouvoir d'expression des logiques, que les langages reconnus par des réseaux de Petri sont strictement inclus dans $NTime[n]$ (la classe des langages reconnaissables par une machine de Turing non-déterministe en temps linéaire). On utilisera la caractérisation logique des langages des réseaux de Petri donnée par Parigot et Pelz [PP85] et une caractérisation de $NTime[n]$ énoncée par Lautemann, Schwentick et Schweikhardt [LSS99].

Dans le chapitre 5, on donne quelques caractérisations logiques de la classe des langages rudimentaires. Bien qu'une caractérisation de cette classe existe, nos caractérisations sont toutes nouvelles et n'utilisent pas les opérarions arithmétiques (sauf l'ordre) dans leurs définitions.

Un des intérêts de ces caractérisations est le jeu, introduit dans le chapitre 6, qu'elles inspirent. On donnera aussi un résultat utilisant la partie existentielle de ce jeu sur la définissabilité de la connexité des graphes finis dans la continuité des travaux de Fagin, de Rougemont et Schwentick.

Dans cette thèse, les résultats propres sont en majuscules, *i. e.* notés "THÉORÈME".

# Chapitre 2

# Préliminaires

## 2.1 Les langages de la hiérarchie de Chomsky

Les *langages* ont été conçus dans le but de communiquer avec les autres personnes, ils sont complets dans la mesure où ils peuvent tout exprimer. Ces langages sont appelés les *langages naturels*. Par la suite, on a considéré des langages *précis* pour s'exprimer dans tel ou tel domaine bien déterminé du savoir. Depuis le dévelopement de l'informatique, la théorie des langages a connu un grand intérêt pour les théoriciens. On va dans cette section définir les langages d'une façon formelle, voir par exemple le livre [Har78] pour plus de détails et de définitions.

**Définition 2.1** *Un* alphabet, ou vocabulaire, *est un ensemble fini non vide de* symboles*, dits aussi* lettres*. Les vocabulaires seront notés, tout au long de cette thèse, par des lettres grecques majuscules. Un* mot *$u$ sur un vocabulaire $\Sigma$ est une suite finie, éventuellement vide, de symboles de $\Sigma$. On notera $|w|$ la* longueur *du mot $w$. On notera $\Sigma^*$ l'ensemble de tous les mots, et $\varepsilon$ le mot vide. Tout sous-ensemble de $\Sigma^*$ est dit un* langage sur $\Sigma$*.*

**Exemple.** L'alphabet des notations binaires est : $\Sigma = \{0, 1\}$.
Un exemple de mots sur $\Sigma$ est 1001, on note $1001 \in \Sigma^*$, avec $|1001| = 4$.
L'ensemble des notations binaires des entiers naturels est $\Sigma^*$.
Les représentations ne sont pas uniques, car 1 et 01 représentent le même entier.
L'ensemble des représentations binaires des entiers pairs est un langage sur $\Sigma$.

$\square$*Exemple*

Ayant défini les langages, on a cherché à les classifier selon la difficulté à les définir ou à les reconnaîtres. Les deux classes, langages finis et langages infinis, constituent une première hiérarchie. Si on prend le point de vue des langages naturels, il y a deux possibilités de classer les langages en fonction de leurs difficultés :

– le point de vue du *locuteur*, le problème est alors de savoir comment engendrer les phrases. Cela conduit à la notion de *grammaire* ;
– le point de vue de *l'auditeur*, le problème est alors de savoir reconnaître les mots du langage. Cela conduit à diverses notions de *reconnaisseurs*.

### 2.1.1 Les langages réguliers

Un des types de reconnaisseurs les plus restrictifs qui a été introduit est celui *d'automate fini*. La finitude est présente à tous les niveaux de la définition de ce modèle.

**Définition 2.2** *Un* automate fini non-déterministe*, noté AFN, est un quintuplet* $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$*, où :*
– $\Sigma$ *est* l'alphabet d'entrée *;*
– $Q$ *est un ensemble fini* d'états *;*
– $F$ *est un sous-ensemble de* $Q$ *des* états finaux *(ou* acceptants*) ;*
– $q_0 \in Q$ *est* l'état initial *;*
– $\delta$ *est la* fonction de transition*, à savoir une application*

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \longrightarrow 2^Q.$$

*Un* automate fini déterministe *AFD, est un AFN dont la fonction de transition vérifie :*

1. *pour tout* $q \in Q$, $\delta(q, \varepsilon) = \emptyset$ *;*
2. *pour tout* $(q, a) \in Q \times \Sigma$, *l'image* $\delta(q, a)$ *de* $(q, a)$ *est un singleton.*

On peut canoniquement étendre $\delta$ en une application :

$$\delta^* : Q \times \Sigma^* \longrightarrow 2^Q;$$

de la façon suivante :
– On définit la $\varepsilon$-clôture d'un état $q$ par induction :

$$Clo_0(q) = \{q\}$$

et :

$$Clo_{i+1}(q) = Clo_i(q) \cup \{p \in Q | \exists s \in Clo_i(p), q \in \delta(s, \varepsilon)\}.$$

On définit alors :

$$closure(q) = \bigcup_{i \in \mathbb{N}} Clo_i(q).$$

– La fonction de transition étendue est définie inductivement par :

$$\delta^*(p, \varepsilon) = closure(\{p\}),$$

$$\delta^*(p, ua) = closure(\bigcup_{s \in \delta^*(p,u)} \delta(s, a)).$$

**Remarque.** La $\varepsilon$-clôture n'a pas d'intérêt pour les automates déterministes. □

**Définition 2.3** *Le langage* $L(\mathcal{A})$ accepté *(on dit aussi* reconnu*) par un AFN* $\mathcal{A}$ *est l'ensemble :*

$$\{w \in \Sigma^* | \delta^*(q_0, w) \cap F \neq \emptyset\}.$$

*Un langage est dit* régulier *s'il est accepté par un AFN. La classe des langages réguliers est notée Reg.*

**Exemple.** Soit le langage $L$ des mots de longueur paire sur $\Sigma = \{0, 1\}$. L'automate $\mathcal{A} = (Q, \Sigma, F, q_0, \delta)$ tel que :
– $Q = \{q_0, q_1\}$ ;
– $F = q_0$ ;
– $\delta(q_0, 0) = \delta(q_0, 1) = \{q_1\}$ et $\delta(q_1, 0) = \delta(q_1, 1) = \{q_0\}$ ;
reconnaît $L$. Donc $L$ est un langage régulier. □*Exemple*

### 2.1.2 Les langages algébriques

Après avoir donné un exemple de reconnaisseurs, on va donner maintenant l'autre point de vue, celui de grammaires. À titre d'illustration, on va définir les langages *algébriques*.

**Définition 2.4** *Une* grammaire algébrique *est un quadruplet* $(\Sigma, N, S, P)$ *tel que :*
– $\Sigma$ *et* $N$ *sont deux ensembles finis disjoints, appelés respectivement l'ensemble des* terminaux *et celui des* non-terminaux *;*
– $S$ *est un symbole spécial de* $N$*, appelé l'*axiome *de* $G$ *;*
– $P$ *est un* ensemble de productions *de la forme* $X \to u$*, où* $X \in N$ *et* $u \in (\Sigma \cup N)^*$.
*On définit la* règle de dérivation $\Rightarrow_G$ *pour une grammaire* $G$ *par :*

$$u_1 X u_2 \Rightarrow_G u_1 u u_2 \text{ est une dérivation ssi } X \to u \in P.$$

*La clôture réflexive et transitive de* $\Rightarrow_G$ *est notée* $\overset{*}{\Rightarrow}_G$.

*Un langage* $L \subseteq \Sigma^*$ *est* algébrique*, noté CFL (pour l'anglais "Context Free Language"), si, et seulement si, il existe une grammaire algébrique* $G$ *qui le dérive à partir de* $S$ *:*

$$L = L(G) = \{w \in \Sigma^* | S \overset{*}{\Rightarrow}_G w\}.$$

*Un* arbre de dérivation d'un mot $w \in L(G)$ *est un arbre étiqueté par* $\Sigma \cup N$ *tel que :*
- *La racine est étiquetée par l'axiome* $S$ *;*
- *les feuilles sont étiquetées par des terminaux ;*
- *les nœuds internes sont étiquetés par des non-terminaux ;*
- *le passage d'un nœud interne à ses fils correspond à une production en respectant leur ordre d'apparition de gauche à droite ;*
- *la lecture des feuilles de gauche à droite donne* $w$.

On va maintenant définir quelques restrictions sur les grammaires algébriques. Ces restrictions vont donner lieu à deux sous-classes célèbres de CFL.

**Définition 2.5** *Une grammaire est dite* linéaire à droite *si toutes les productions sont de l'une des formes suivantes :*

1. $A \to aB$ *;*
2. $A \to a$ *;*
3. $A \to \varepsilon$ *;*

*pour* $A, B \in N$*, et* $a \in \Sigma$.

**Définition 2.6** *Une grammaire algébrique est dite* non-ambiguë *si chaque mot de* $L(G)$ *admet un unique arbre de dérivation. Un langage algébrique* $L$ *est* non-ambigu*, noté* $UCFL$ *(pour l'anglais* unambiguous context free language*), s'il existe une grammaire* $G$ *non-ambiguë, telle que* $L = L(G)$.

On va maintenant énoncer un résultat qui relie les langages réguliers aux grammaires algébriques.

**Proposition 2.1** *Un langage* $L$ *est régulier si, et seulement si, il existe une grammaire* linéaire à droite*,* $G$*, telle que* $L = L(G)$.

On en conclut que les langages réguliers sont des langages algébriques. Ils sont même non-ambigus. De même que les langages réguliers, les langages algébriques ont aussi un modèle de reconnaissance, appelé *automate à pile non déterministe*, qui étend naturellement les automates finis.

### 2.1.3 La hiérarchie de Chomsky

Les langages algèbriques et les langages réguliers font partie de la *hiérarchie de Chomsky*. Les quatres classes constituant cette hiérarchie sont rappelées dans le tableau suivant :

| *Langages* | *Classe de Chomsky* |
|---|---|
| Calculables | Type 0 |
| Sensibles au contexte | Type 1 |
| Algébriques | Type 2 |
| Réguliers | Type 3 |

avec les notations suivantes :

**Type 0** "aucune restriction sur les productions".

**Type 1** "restriction sur les productions : $\alpha \rightarrow \beta$ tel que $|\beta| \geq |\alpha|$" : grammaires sensibles au contexte, on notera CS (pour *context sensitive* en anglais).

**Type 2** "restriction sur les productions : $A \rightarrow \alpha$" : grammaires algébriques, ou indépendantes du contexte CF (*Context Free* en anglais).

**Type 3** "restriction sur les productions : $A \rightarrow a|aB$" : grammaires linéaires droite.

## 2.2 Les langages des réseaux de Petri

Les réseaux de Petri ont été introduits dans le but d'étudier la concurrence. Il s'agit à l'origine d'un modèle de parallélisme. Ce modèle a aussi été étudié comme reconnaisseur, donc comme modèle séquentiel. C'est ce dernier point de vue qui nous intéressera dans ce travail. Dans cette section on va donner quelques définitions des réseaux de Petri, et quelques résultats sur les langages reconnus par des réseaux de Petri, notés $PNL$. Pour une introduction plus détaillée, et pour les motivations qui ont conduits à l'étude du comportement séquentiel des réseaux de Petri, voir, par exemple, le chapitre 6 du livre [Pet81].

**Définition 2.7** *1. Un* réseau de Petri *est un quadruplet $R = (P, T, f, b)$ où $P = \{p_1, \ldots, p_r\}$ est l'ensemble (fini) des* places*, $T = \{t_1, \ldots, t_s\}$ est l'ensemble (fini) des* transitions*, $f : P \times T \longrightarrow \mathbb{N}$ est la* fonction d'incidence avant*, et $b : P \times T \longrightarrow \mathbb{N}$ la* fonction d'incidence arrière.

*2. Un* marquage *du réseau de Petri $R$ est une fonction $M : P \longrightarrow \mathbb{N}$ ; si $M(p) = i$, on dit que la place $p$ contient $i$* jetons*.*

*3. Une transition $t$ est dite* franchissable *à un marquage $M$ si, et seulement si, pour chaque place $p \in P$ on a $M(p) \geq f(p,t)$. On notera alors $M(t\rangle$.*

*4. Si une transition $t$ est franchissable à un marquage $M$, on peut passer du marquage $M$ au marquage $M'$ par $t$ comme suit : $t$ prend de chaque place $p \in P$, $f(p,t)$ jetons et lui rajoute $b(p,t)$ jetons. Le marquage résultant, $M'$, est tel que, pour tout $p \in P$, on a $M'(p) = M(p) - f(p,t) + b(p,t)$, ce qu'on écrit $M(t\rangle M'$.*

*5. On peut ordonner les marquages par :*

$$M_1 \leq M_2 \text{ ssi, pour tout } i \leq r, M_1(p_i) \leq M_2(p_i).$$

*6. Une suite $\overline{t} = (t_1, \ldots, t_n)$ de transitions est franchissable au marquage $M_1$ s'il existe des marquages $M_2, \ldots, M_{n+1}$ tels que pour tout $i \leq n$, $M_i(t_i\rangle M_{i+1}$ ; on note aussi $M_1(\overline{t}\rangle M_{n+1}$.*

**Notations.**

1. On appelle $A$ l'ensemble des *arcs* de $R$ défini par $A = \{(p,t)|f(p,t) > 0\} \cup \{(t,p)|b(p,t) > 0\}$, et on écrit $p \rightarrow^{f(p,t)} t$ ou $t \rightarrow^{b(p,t)} p$ pour les éléments de $A$. Lorsque $f(p,t)$ ou $b(p,t)$ est 1, on omet la valuation de l'arc.
2. Pour une place $p \in P$ donnée, l'ensemble $I(p)$ des *transitions d'entrée* est $\{t|(p,t) \in A\}$ et l'ensemble des *transitions de sortie* $O(p)$ est $\{t|(t,p) \in A\}$.

On peut associer certains langages sur un alphabet $\Sigma$ à des réseaux de Petri en étiquetant les transitions par les symboles de $\Sigma$ (et non avec des mots, comme on le fait d'habitude) et en spécifiant un marquage initial et des marquages finaux. Précisons cette façon de faire dans la définition suivante.

**Définition 2.8** *1- Un* réseau de Petri étiqueté sans $\varepsilon$ *est un triplet constitué d'un réseau de Petri $R$ avec un étiquetage $e : T \longrightarrow \Sigma$, de $T$ sur l'alphabet $\Sigma$, d'un marquage de $P$, appelé* marquage initial*, noté $M_0$, et d'un ensemble fini de marquages, dits* marquages finaux*, $F$.*

*2- Le* langage associé au réseau de Petri $R = (P, T, f, b, e, M_0, F)$ *est défini par :*

$$L(R) = \{w \in \Sigma^+ | \exists M \in F \exists \bar{t} \in T^+ (M_0(\bar{t}\rangle M) \text{ et } e(\bar{t}) = w\}.$$

*3- On note PNL la classe des langages reconnus par des réseaux de Petri.*

Ce modèle de reconnaissance a été comparé aux modèles de calcul classiques de la hiérarchie de Chomsky. Il a été prouvé que :

$$Reg \subsetneq PNL \subsetneq CS,$$

voir par exemple le chapitre 6 du livre [Pet81] pour des preuves détaillées.

## 2.3 Théorie des modèles classique

La *logique mathématique* a comme objectif premier de formaliser le raisonnement mathématique. Les principales investigations de la logique mathématique portent sur les relations entre *les structures, les langages* et *les preuves*.

La *théorie des modèles* est la branche de la logique mathématique qui se rapporte à l'étude des liens entre les structures et les langages. Elle s'intéresse, en particulier, aux problèmes de définissabilité. Elle pose des questions comme :

- Quelles sont les propriétés définissables (ou exprimables) dans tel langage de la logique (du premier ordre) ?
- Quelles sont les structures et relations définissables dans ce langage logique ?
- Quel est le pouvoir d'expression des différents langages logiques ?

Dans cette section on va fixer les notations de théorie des modèles et de la logique du premier ordre qu'on va utiliser tout au long de cette thèse. On va commencer par définir le langage de la logique du premier ordre, ensuite rappeler les notions de structure sur une signature, de formules, de modèles. Pour plus de details voir, par exemple, [CL93, LdR93, CK90].

### 2.3.1 Structures

**Définition 2.9** *Un* type de similarité *ou* signature[1] *relationnel*

$$\tau = (R_1^{a_1}, \cdots, R_n^{a_n}, c_1, \cdots, c_m)$$

*est un uplet constitué de symboles de relations (ou prédicats), et de symboles de constantes. Les exposants $a_i$ sont des entiers représentant l'arité des symboles correspondants.*

La définition générale des signatures contient aussi des symboles de fonctions. On peut toujours identifier les fonctions à des relations qui représentent leurs graphes. On ne va utiliser dans ce travail que des signatures relationnelles, *i.e.* des types de similarité qui ne contiennent pas de symboles de fonction.

**Exemple.**

1. La signature de la théorie des graphes est, en général, $\tau_G = (E^2)$, la relation binaire $E$ étant la relation d'accessibilité du graphe.
2. La signature de la théorie des corps peut être $\tau_C = (0, 1, +^3, \times^3)$. Les symboles $+^3$ et $\times^3$ sont des relations ternaires qui représentent, respectivement le graphe de l'addition et celui de la multiplication.

□*Exemple*

**Définition 2.10** *1. Une* structure *de signature $\tau$, appelée $\tau$-*structure*, est un uplet :*

$$\mathcal{A} = (A, R_1^{\mathcal{A}}, \cdots, R_n^{\mathcal{A}}, c_1^{\mathcal{A}}, \cdots, c_r^{\mathcal{A}})$$

*où $A$ est un ensemble non vide appelé* univers *ou* domaine*. À chaque symbole de relation $R_i^{a_i} \in \tau$ on associe un ensemble $R_i^{\mathcal{A}} \subseteq A^{a_i}$. À chaque symbole de constante $c_i \in \tau$ on associe un élément distingué $c_i^{\mathcal{A}} \in A$. On omet, s'il n'y a aucune confusion possible, l'exposant $\mathcal{A}$.*

*2. Soient $\mathcal{A}$ et $\mathcal{B}$ deux $\tau$-structures. On dit que $\mathcal{A}$ est une* sous-structure *de $\mathcal{B}$ si l'univers de $\mathcal{A}$ est inclus dans celui de $\mathcal{B}$ et que $c^{\mathcal{A}} = c^{\mathcal{B}}$ pour tout symbole de constante $c \in \tau$, et si pour tout $(a_1, \cdots, a_n) \in A^n$ et toute relation $R \in \tau$ d'arité $n$, on a :*

$$(a_1, \cdots, a_n) \in R^{\mathcal{A}} \Leftrightarrow (a_1, \cdots, a_n) \in R^{\mathcal{B}}.$$

[1] Habituellement appelé *langage ou vocabulaire*, mais on réserve ces termes pour la terminologie de la théorie des langages formels.

**Exemple.** Soit $\mathbb{R}$ l'ensemble des nombres réels. Alors $(\mathbb{R}, 0, 1, +, \times)$ est une $\tau_C$-structure. L'interprétation de 0 et 1 sont les réels zéro et un ; $+$ est vue comme une relation ternaire représentant le graphe de l'addition sur les réels : $+(x, y, z)$ ssi $x + y = z$ ; de même pour $\times$ on a $\times(x, y, z)$ ssi $x \times y = z$. On peut vérifier que $(\mathbb{N}, 0, 1, +, \times)$ est une sous-structure de $(\mathbb{R}, 0, 1, +, \times)$. $\square$*Exemple*

### 2.3.2 Logique du premier ordre

On commence par définir la syntaxe de la logique du premier ordre $FO$. Soit

$$\tau = (R_1^{a_1}, \cdots, R_n^{a_n}, c_1, \cdots, c_m)$$

une signature relationnelle. Toute formule du langage de la logique du premier ordre de $\tau$, noté $FO(\tau)$, est déjà un mot de l'alphabet, infini pour ce cas, constitué de :
- $v_0, v_1, \cdots$ (les variables) ;
- $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ (les connecteurs) ;
- $\exists, \forall$ (les quantificateurs) ;
- $=$ (on n'utilisera que des langages égalitaires) ;
- ),( ;
- et les symboles de $\tau$.

**Définition 2.11** *1. On appellera* terme *de la signature $\tau$ les variables et les constantes de $\tau$*[2].

*2. Les* formules du langage de la logique du premier ordre de $\tau$, *dites aussi $FO(\tau)$-formules, sont construites inductivement comme suit :*

*(a) si $x$ et $y$ sont des termes alors $x = y$ est une $FO(\tau)$-formule ;*

*(b) si $R^a \in \tau$ et si $x_1, \cdots, x_a$ sont des termes alors $R(x_1, \cdots, x_a)$ est une $FO(\tau)$-formule ;*

*(c) si $\varphi$ est une $FO(\tau)$-formule alors $\neg\varphi$ est une $FO(\tau)$-formule ;*

*(d) si $\varphi, \psi$ sont des $FO(\tau)$-formules et $\circ \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$ alors $(\varphi \circ \psi)$ est une $FO(\tau)$-formule ;*

*(e) si $\varphi$ est une $FO(\tau)$-formule et si $x$ une variable alors $\exists x\varphi$ et $\forall x\varphi$ sont des $FO(\tau)$-formules.*

*3. On appellera* formule atomique *toute formule définie à l'aide des règles (a) et (b) uniquement.*

On va maintenant donner des extensions de la logique du premier ordre qu'on va utiliser ultérieurement.

[2]On rappelle que nos signatures ne contiennent pas de symboles de fonction.

### 2.3.3 Logique du second ordre

**Définition 2.12** *1. L'alphabet de* la logique du second ordre *est l'alphabet du premier ordre, auquel on rajoute un ensemble dénombrable de symboles de* variables du second ordre, *dotée chacune d'une arité, qu'on notera par des lettres majuscules.*

*2. L'ensemble des variables du second ordre sera noté $\mathcal{V}$.*

*3.* On définit l'ensemble des formules du second ordre sur la signature $\tau$, *noté $SO(\tau)$, inductivement par :*

*(a) Toute formule de $FO(\tau \cup \mathcal{V})$ est une formule de $SO(\tau)$.*

*(b) Soient $\varphi$ et $\psi$ deux formules de $SO(\tau)$ et $x$ une variable individuelle, alors $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \rightarrow \psi, \varphi \leftrightarrow \psi, \exists x\varphi, \forall x\psi$ sont des formules de $SO(\tau)$.*

*(c) Soient $\varphi$ une formule de $SO(\tau)$, $X \in \mathcal{V}$ une variable de relation $n$-aire et $x_1, \cdots, x_n$ des termes, alors $X(x_1, \cdots, x_n), \forall X\varphi$ et $\exists X\varphi$ sont des formules de $SO(\tau)$.*

**Définition 2.13** *Une occurrence d'une variable est dite* liée *si elle est dans le champs d'un quantificateur, sinon elle est dite* libre. *Si une variable a au moins une occurrence libre dans une formule, on dit qu'elle est* libre *dans la formule. On écrira $\varphi(X_1, \cdots, X_s, v_1, \cdots, v_n)$ pour dire que les variables libres de $\varphi$ sont parmi $X_1, \cdots, X_s$, $v_1, \cdots, v_n$.*

*On appelle* formule close *ou* énoncé *toute formule sans variables libres.*

**Exemple.** Dans la formule :

$$\exists R^2 \forall x(R(x, y) \vee \exists x \forall y S(x, y)),$$

les occurrences de la variable $x$ sont toutes les deux liées, par contre la première occurrence de la variable $y$ est libre et la seconde est liée. Donc $y$ est libre dans cette formule ainsi que la variable $S$. $\square$*Exemple*

### 2.3.4 Sémantique

Définissons maintenant les notions d'interprétation et de satisfaction d'une formule dans une structure.

**Définition 2.14** *Soit $\mathcal{A}$ une $\tau$-structure. Une* interprétation *des variables dans $\mathcal{A}$ est une application*

$$\underline{int} : V \cup \mathcal{V} \longrightarrow A\,,$$

*qui à chaque variable individuelle $x$ fait correspondre un élément de $A$, et à chaque variable du second ordre $X$ fait correspondre un sous-ensemble de $A^{ar(X)}$, où $ar(X)$ représente l'arité de $X$. On peut étendre $\underline{int}$ aux constantes de la signature $\tau$ en imposant $\underline{int}(c) = c^{\mathcal{A}}$.*

La notion de satisfaction d'une formule dans une structure est l'une des notions les plus imporantantes de la théorie des modèles. On appelle cette partie la *sémantique*. C'est l'essence même de la théorie des modèles qui s'intéresse à la relation entre le langage d'une logique et les structures.

**Définition 2.15** *Soit $\varphi(v_1, \cdots, v_n)$ une formule. On notera $(\mathcal{A}, i) \models \varphi$ le fait que la formule $\varphi$ avec l'interprétation $i$ soit satisfaite dans la structure $\mathcal{A}$, ou encore $(\mathcal{A}, i)$ est un modèle de la formule $\varphi$.*

*On définit inductivement si une $\tau$-formule du premier ordre est satisfaite dans $(\mathcal{A}, i)$ par :*

$$\begin{aligned}
(\mathcal{A}, i) \models t_1 = t_2 &\Leftrightarrow i(t_1) = i(t_2)\\
(\mathcal{A}, i) \models R_j(t_1, \cdots, t_{a_j}) &\Leftrightarrow (i(t_1), \cdots, i(t_{a_j})) \in R_j^{\mathcal{A}}\\
(\mathcal{A}, i) \models \neg\varphi &\Leftrightarrow (\mathcal{A}, i) \not\models \varphi\\
(\mathcal{A}, i) \models \varphi \wedge \psi &\Leftrightarrow (\mathcal{A}, i) \models \varphi \text{ et } (\mathcal{A}, i) \models \psi\\
(\mathcal{A}, i) \models \varphi \vee \psi &\Leftrightarrow (\mathcal{A}, i) \models \varphi \text{ ou } (\mathcal{A}, i) \models \psi\\
(\mathcal{A}, i) \models \varphi \rightarrow \psi &\Leftrightarrow (\mathcal{A}, i) \models (\neg\varphi \vee \psi)\\
(\mathcal{A}, i) \models \varphi \leftrightarrow \psi &\Leftrightarrow (\mathcal{A}, i) \models (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)\\
(\mathcal{A}, i) \models \exists v\varphi &\Leftrightarrow \text{il existe } a \in A \text{ tel que } (\mathcal{A}, i, a/v) \models \varphi\\
(\mathcal{A}, i) \models \forall v\psi &\Leftrightarrow (\mathcal{A}, i) \models \neg\exists v\neg\varphi
\end{aligned}$$

*On remarque que la vérité de $\varphi$ dans $(\mathcal{A}, i)$ ne dépend que des valeurs de $i$ aux variables libres de $\varphi$. Alors les formules closes sont ou bien vraies ou bien fausses dans chaque structure. Pour ce qui est de la sémantique de SO, les interprétations des variables du second ordre sont des sous-ensembles de l'univers de la structure à*

*la puissance l'arité de la variable, et le sens de la formule sera comme celui de FO sur la signature augmentée* $\tau \cup \mathcal{V}$.

*Une formule close est dite* valide *si elle est vraie dans tout modèle.*

**Définition 2.16** *Soit* $\tau$ *une signature logique. Deux* $\tau$*-structures* $\mathcal{A}$ *et* $\mathcal{B}$ *sont dites* élémentairement équivalentes*, on note* $\mathcal{A} \equiv \mathcal{B}$ *si, et seulement si, pour toute formule* $\varphi \in FO(\tau)$ *on a :*

$$\mathcal{A} \models \varphi \Longleftrightarrow \mathcal{B} \models \varphi.$$

Une grande partie de la théorie des modèles classique s'intéresse à l'étude de la relation d'équivalence élémentaire. On passe maintenant à la définition de l'équivalence logique.

**Définition 2.17** *Une formule* $\varphi$ *est* conséquence logique *d'une formule* $\psi$ *si pour toute structure* $\mathcal{A}$ *et pour toute interprétation* $i$ *on a :*

$$(\mathcal{A}, i) \models \psi \text{ alors } (\mathcal{A}, i) \models \varphi$$

*On dit que deux formules* $\varphi$ *et* $\psi$ *sont* logiquement équivalentes *si* $\varphi$ *est conséquence logique de* $\psi$ *et* $\psi$ *est conséquence logique de* $\varphi$.

Il existe une forme normale fort utile pour classifier les ensembles de formules. Cette forme normale est la forme prénexe.

### 2.3.5 La forme prénexe

**Définition 2.18** *Une formule de SO est dite* prénexe *si elle est de la forme :*

$$\bar{Q}\bar{X}\bar{P}\bar{x}\varphi$$

*où les $Q_i \in \bar{Q}$ et $P_j \in \bar{P}$ sont des quantificateurs $\exists$ ou $\forall$, les $X_i$ sont des variables du second ordre, les $x_i$ sont des variables du premier ordre et $\varphi$ est une formule sans quantificateur (dite aussi* formule ouverte*). Si tous les $Q_i$ sont $\exists$, on dit que la* formule est existentielle du second ordre. *La* logique existentielle du second ordre*, notée $\exists SO$, est l'ensemble des formules existentielles du second ordre.*

*Le cas des formules de FO peut être vu comme un cas particulier des formules de SO sans variables du second ordre.*

**Résultat 2.1** *Toute formule du second ordre est logiquement équivalente à une formule sous forme prénexe. De même toute formule de FO est logiquement équivalente à une formule de FO sous forme prénexe.*

**Définition 2.19** *Une formule du second ordre dans laquelle toutes les variables du second ordre sont unaires est dite* monadique*. L'ensemble des formules monadiques est appelé* la logique monadique du second ordre*, notée $MSO$.*

**Résultat 2.2** *Toute formule de $MSO$ est logiquement équivalente à une formule de $MSO$ sous forme prénexe.*

**Preuve :** Pour éliminer les alternances entre les quantificateurs du premier ordre et les quantificateurs du second ordre, on procède de la façon suivante :

$$\forall x \exists Y \phi(x, Y) \equiv \forall X \exists Y \exists u \exists v \forall x((X(u) \wedge X(v) \wedge u \neq v) \vee (X(x) \rightarrow \phi(x, Y)))$$

$$\exists x \exists Y \phi(x, Y) \equiv \exists X \exists Y \exists x(X(x) \wedge \phi(x, Y)) \equiv \exists Y \exists x \phi(x, Y).$$

Le même procédé reste valable pour les quantifications universelles. $\square Preuve$

### 2.3.6 Restrictions sémantiques

Une autre approche qui vient des définitions mathématiques est la notion de *restriction sémantique,* c'est-à-dire une restriction des variables du second ordre à être dans un ensemble de relations de même arité de l'univers.

**Définition 2.20** *Soit $\mathcal{C}$ une classe de relations de même arité, et soit $\mathcal{A}$ une structure. On note $\mathcal{A} \models \exists X \in \mathcal{C}\varphi(X, \cdots)$ si, et seulement si, il existe une relation $R \in \mathcal{C}$ de $X$ qui est aussi dans $\mathcal{C}$, telle que $\mathcal{A} \models \varphi(R, \cdots)$. Par dualité on définit la quantification universelle.*

**Exemple.** Soit $Fonc$ la classe des graphes de fonctions. La formule

$$\exists X \in Fonc\, \forall x \exists y (X(y, x) \wedge \forall z (X(z, x) \rightarrow z = y))$$

dit qu'il existe un graphe d'une fonction bijective. $\square Exemple$

### 2.3.7 Isomorphismes de structures

**Définition 2.21** *Soient $\mathcal{A}$ et $\mathcal{B}$ deux structures sur une même signature $\tau$. Un* isomorphisme *entre $\mathcal{A}$ et $\mathcal{B}$ est une bijection $f$ de l'univers de $\mathcal{A}$ dans l'univers de $\mathcal{B}$ telle que :*

1. *Pour tout symbole de constante $c \in \tau$, on a $f(c^{\mathcal{A}}) = c^{\mathcal{B}}$ ;*
2. *Pour tout symbole de relation $k$-aire $R \in \tau$, et pour tout $k$-uplet $(a_1, \cdots, a_k) \in \mathcal{A}$, on a $(a_1, \cdots, a_k) \in R^{\mathcal{A}} \Leftrightarrow (f(a_1), \cdots, f(a_k)) \in R^{\mathcal{B}}$.*

*S'il existe un isomorphime entre $\mathcal{A}$ et $\mathcal{B}$, on dit que $\mathcal{A}$ et $\mathcal{B}$ sont* isomorphes*, et on note $\mathcal{A} \cong \mathcal{B}$.*

## 2.4 Théorie des modèles finis

Vu que les objets et les méthodes informatiques les plus intéressants (les entrées, les bases de données, les programmes, etc.) sont finis, l'étude des structures finies a fait l'objet d'un grand intérêt de la part des informaticiens. Voici un tableau qui illustre la théorie des modèles en informatique. Rappelons que la théorie des modèles est l'étude des liens entre les structures et les langages.

| Structure | Langage |
|---|---|
| Structures de données | Algorithmes |
| Bases de données | Langages de requêtes |
| ⋮ | ⋮ |

Pour une plus ample introduction à la théorie des modèles finis, voir par exemple [LdR96, EF95, Imm99, Gur84].

**Définition 2.22** *La* théorie des modèles finis *est l'étude des structures dont le domaine est fini.*

La théorie des modèles classique a émergé à partir des paradoxes de l'infini et du désir de comprendre ce dernier. La construction des modèles a longtemps été basée sur l'infini, et la plupart des outils et des méthodes utilisait fortement l'infinité de ces structures. Dans ce contexte, l'étude des structures finies était le domaine de quelques rares spécialistes. L'essor de l'informatique théorique a été bénéfique pour cette discipline.

### 2.4.1 Logique du premier ordre sur les structures finies

Je voudrai commencer cette section par rappeler une phrase de Chang et Keisler dans leur célèbre livre *"model theory"* :

> *"In almost all the deeper theorems in model theory the key of the proof is to construct the right kind of model* $\cdots$*"* [CK90, page 2]

On remarquera, vu le nombre de propriétés qui ne sont plus vérifiée quand on se restreint aux seules structures finies, que la garantie de l'existance de tels modèles est leur caractère infini.

On va énoncer dans cette section quelques propriétés célèbres de la théorie des modèles classique qui ne se généralisent pas lorsqu'on se restreint aux seules structures finies, pour plus de détails voir par exemple [EF95, Gur84]. La première propriété est celle de *compacité*, dite aussi de *finitude.*

**Théorème 2.1 (Compacité)** *Soit* $\Phi$ *une théorie, ensemble de formules du premier ordre. Si tout sous-ensemble fini de* $\Phi$ *a un modèle alors* $\Phi$ *a un modèle (fini ou infini).*

Ce théorème est un des critères de satisfiabilité les plus utilisés en théorie des modèles classique. Le théorème suivant dit que la satisfiabilité des sous-théories finies n'est plus un critère de satisfiabilité sur les structures finies.

**Théorème 2.2** *Il existe un ensemble* $T$ *d'énoncés du premier ordre dont tout sous-ensemble fini admet un modèle fini, mais* $T$ *lui-même n'a pas de modèle fini.*

**Preuve :**

Pour tout entier naturel $n$, soit la formule :

$$\varphi_n \equiv \exists x_1 \cdots x_n \bigwedge_{1 \leq i < j \leq n} x_i \neq x_j.$$

Pour un $n$ donné, $\mathcal{A} \models \varphi_n$ si, et seulement si, l'univers de $\mathcal{A}$ contient au moins $n$ éléments. Soit

$$T = \{\varphi_n | n \in \omega\},$$

évidemment cet ensemble de formules n'admet pas de modèle fini. Par contre, si on prend un sous ensemble fini de $S \subseteq T$ il sera inclus dans

$$T_k = \{\varphi_n | n < k\},$$

pour un certain $k \in \omega$. Il nous suffit donc de prendre un modèle à $k$ éléments et on a un un modèle de $T_k$, et donc de $S$. $\square Preuve$

Une autre propriété qui ne résiste pas au passage des modèles généraux (finis et infinis) aux seuls modèles finis est celle de *complétude*, propriété qui est pourtant l'un des piliers de la théorie des modèles classique. Cette propriété permet, dans le cadre de la théorie des modèles classiques, de relier conséquence syntaxique et conséquence sémantique. Le théorème de complétude peut s'énoncer comme suit :

**Théorème 2.3 (Complétude)** *L'ensemble des formules closes valides de la logique du premier ordre est récursivement énumérable.*

Trakhtenbrot a prouvé que pour les seuls modèles finis, ceci n'est plus le cas, voir par exemple [EF95, Gur84, LdR96], pour une preuve détaillée.

**Théorème 2.4 (Trakhtenbrot)** *L'ensemble $Val$ de tous les énoncés vrais dans tous les modèles finis n'est pas récursivement énumérable.*

Une conséquence directe est :

**Corollaire 2.1** *Une axiomatisation effective de $FO$ sur les modèles finis n'est pas possible.*

**Preuve :**(du théorème)

Le comportement des machines de Turing peut être décrit par des formules du premier ordre sur une signature appropriée $\sigma$. En particulier, pour toute machine de Turing $M$, il existe un énoncé $\phi_M \in FO[\sigma]$ tel que $M$ s'arrête si, et seulement si, $\phi_M$ admet un modèle fini. Puisque l'ensemble des machines qui ne s'arrêtent pas n'est pas récursivement énumérable, l'ensemble

$$V = \{\phi | \neg\phi \text{ n'a pas de modèle fini } \}$$

n'est pas récursivement énumérable. C'est une réduction à un célèbre problème indécidable, qui est le problème de la *Halte* des machines de Turing (en terminologie anglo-saxonne *The Halting Problem*). $\square Preuve$

De plus, le fait que chaque modèle fini est caractérisé, à isomorphisme près, par une formule du premier ordre, voir par exemple [EF95], nous montre que l'étude des théories complètes dans le cadre des modèles finis est triviale.

**Théorème 2.5** *Pour toute signature $\tau$, pour toute $\tau$-structure finie $\mathcal{A}$, il existe un énoncé $\psi_{\mathcal{A}} \in FO(\tau)$ tel que pour chaque $\tau$-structure $\mathcal{B}$ :*

$$\mathcal{B} \models \psi_{\mathcal{A}} \Leftrightarrow \mathcal{A} \cong \mathcal{B}.$$

**Preuve :**

Soit $\mathcal{A}$ une $\tau$-structure dont l'univers est $\{a_1, \cdots, a_n\}$. On prend comme formule $\varphi_{\mathcal{A}}$, la formule formée de la conjonction de :
– $\phi(x_1, \cdots, x_n)$ pour toute formule atomique $\phi$ telle que :

$$\mathcal{A} \models \phi(a_1, \cdots, a_n)\ ;$$

– $\neg\phi(x_1, \cdots, x_n)$ pour toute formule atomique $\phi$ telle que :

$$\mathcal{A} \not\models \phi(a_1, \cdots, a_n)\ ;$$

– et enfin la formule qui dit qu'il existe exactement $n$ éléments dans l'univers de la structure.
La formule

$$\psi_{\mathcal{A}} \equiv \exists x_1 \cdots x_n \varphi_{\mathcal{A}}.$$

vérifie les conditions du théorème, voir [EF95]. $\square Preuve$

Comme conséquence de ce résultat, on perd la branche de la théorie des modèles qui s'intéresse à l'étude de la relation d'équivalence élémentaire entre les structures d'une même signature.

Toute classe de structures finies admet une théorie dont les seuls modèles sont les structures de cette classe.

**Corollaire 2.2** *Pour toute classe $\mathcal{K}$ de structures logiques finies, close par isomorphisme, de même signature $\sigma$, il existe un ensemble de formules closes du premier ordre, $T$, tel que pour toute structure $\mathcal{A}$ :*

$$\mathcal{A} \in \mathcal{K} \Leftrightarrow \mathcal{A} \models T.$$

**Preuve :**

Pour cela il suffit de prendre

$$T = \{\neg\psi_{\mathcal{B}} | \mathcal{B} \notin \mathcal{K}\}.$$

Avec $\psi_{\mathcal{B}}$ est la formule caractérisant la structure $\mathcal{B}$ à isomorphisme près, comme dans le théorème précédent. $\square Preuve$

### 2.4.2 Définitions implicites et résultat de Beth

Intuitivement, une formule est une *définition explicite* d'une relation $R$ si elle décrit une condition que doivent satisfaire les uplets d'éléments pour être dans $R$ ; par contre, une *définition implicite* est une formule que doit satisfaire la relation $R$ elle-même.

**Exemple.** Le prédicat $P(x)$ suivant :

$$P(0) \wedge \forall x(P(x) \leftrightarrow \neg P(x+1)).$$

dit implicitement que $x$ est dans $P$ si, et seulemnt si, c'est une position paire dans les segments initiaux de $\mathbb{N}$ munis de l'ordre induit. $\square Exemple$

**Définition 2.23** *Soit $\tau$ une signature logique et $\mathcal{K}$ une classe de $\tau$-structures. Soit $R$ un symbole de relation $k$-aire qui n'est pas dans $\tau$.*

1. *On dit que $R$ est* explicitement *définie par une formule $\phi$ du premier ordre sur $\mathcal{K}$ si pour toute $\tau$-structure $\mathcal{A} \in \mathcal{K}$,*

$$R^{\mathcal{A}} = \{x_1, \ldots, x_k \in |\mathcal{A}| : \mathcal{A} \models \phi(x_1, \ldots, x_k)\}.$$

2. *Une $\tau \cup \{R\}$-formule du premier ordre, $\phi(R)$,* définit $R$ implicitement *sur $\mathcal{K}$ si chaque $\tau$-structure de $\mathcal{K}$ admet au plus une expansion en une $\tau \cup \{R\}$-structure satisfaisant $\phi(R)$.*

Il est facile de vérifier que toute relation définissable explicitement est définissable implicitement.

**Proposition 2.2** *Toute relation explicitement définissable est implicitement définissable.*

**Preuve :**

Soit $\phi$ la formule qui définit explicitement une relation $k$-aire, $R$, sur les $\tau$-structures. Alors la formule $\Psi(S)$ sur $\tau \cup \{S\}$, où $S$ est un symbole de relation $k$-aire, définie par $\Psi(S) \Leftrightarrow \forall x_1 \ldots x_k(S(x_1, \ldots, x_k) \leftrightarrow \phi(x_1, \ldots, x_k))$ est une définition implicite de la relation $R$. $\square Preuve$

#### Cas des structures sans restrictions de cardinalité

Le résultat précédent est la partie facile du célèbre théorème de Beth, voir par exemple l'ouvrage [CL93] pour plus de détails.

**Théorème 2.6 (Beth, 1953)** *Sur la classe de toutes les structures (finies et infinies), une relation est implicitement définissable au premier ordre si, et seulement si, elle l'est explicitement.*

Une preuve classique de ce théorème utilise le lemme d'interpolation de Craig, qui lui-même utilise le théorème de compacité. Le résultat de Beth eut comme conséquence le manque d'intérêt des logiciens pour les définitions implicites en elles-mêmes. L'intérêt pour les définitions implicites a ressurgi avec l'essor de la théorie des modèles finis.

### Cas des structures finies

En remarquant que la propriété de compacité n'est plus vraie si on se restreint aux modèles finis, Gurevich s'est intéressé au statut des conséquences de cette propriété en théorie des modèles classiques dans le cas des modèles finis. Il a prouvé, dans [Gur84], que la propriété d'interpolation de Craig, la propriété de préservation de Lós-Tarski et d'autres propriétés ne sont plus vraies dans le cadre de la théorie des modèles finis. Ce qui nous intéresse dans cette partie est le théorème qui suit.

**Théorème 2.7 (Gurevich, [Gur84])** *Si on se restreint aux structures finies, il existe une relation qui est implicitement définie et qui n'est pas explicitement définissable.*

**Preuve :** (Idée)

La relation utilisée dans la preuve de ce résultat dans [Gur84] est : le prédicat *parité sur les ordres linéaires.* Cette propriété n'est pas explicitement définissable dans la logique du premier ordre, comme on le verra à l'aide des jeux d'Ehrenfeucht-Fraïssé. Gurevich a prouvé qu'elle est définissable implicitement. La formule qui définit implicitement ce prédicat est la conjonction des quatres conditions suivantes :

1. $<$ est un ordre discret ;
2. le premier élément, par rapport à $<$, est dans $P$ ;
3. pour chaque élément $x$, on a : $x \in P$ si, et seulement si, son prédécesseur n'est pas dans $P$ ;
4. le dernier élément, par rapport à $<$, n'est pas dans $P$.

$\square Preuve$

On va dans le chapitre 3 donner une nouvelle preuve de ce résultat qui est conséquence d'un résultat de la théorie des langages.

### 2.4.3 Jeux de $FO$ et jeux de $MSO$ sur les structures finies

Parmi les rares outils hérités par la théorie des modèles finis de la théorie des modèles classiques on trouve les *jeux d'Ehrenfeucht-Fraïssé*, et les diffèrentes notions de localité de la logique du premier ordre. D'autres outils sont propres aux modèles finis, comme les lois 0-1. Les jeux d'Ehrenfeucht-Fraïssé, et leurs variantes, restent un outil pour prouver qu'une propriété quelconque n'est pas définissable par une formule de la logique correspondant à ces jeux. Les problèmes de non-définissabilité restent parmi les problèmes les plus difficiles de la théorie des modèles finis, ceci est dû à leurs liens intimes avec des problèmes de recherche de bornes inférieures pour des problèmes en complexité structurelle. Pour plus de détails voir par exemple [EF95, Imm99, FSV95, Sch96]

#### Cas du premier ordre

**Définition 2.24** *Soit $\phi$ une formule du premier ordre sous forme prénexe. On appelle* rang de quantification de $\phi$, *et on notera $rq(\phi)$, le nombre de quantificateurs dans $\phi$.*

**Exemple.** Le rang de quantification de la formule

$$\forall x \exists y (x < y)$$

est 2. $\square Exemple$

On voit, par définition, que le rang de quantification des formules atomiques est 0.

**Définition 2.25** *Soient $\mathcal{A}$ et $\mathcal{B}$ deux structures sur une signature $\tau$. On notera*

$$\mathcal{A} \equiv_k \mathcal{B},$$

*le fait que $\mathcal{A}$ et $\mathcal{B}$ satisfont les mêmes formules de rang de quantification au plus $k$.*

Supposons qu'on veuille affirmer que la structure $\mathcal{A}$ satisfait la formule $\exists x \phi(x)$. On choisit alors un élément du domaine de $\mathcal{A}$ et on vérifie que cet élément satisfait $\phi$. Si, par contre, on veut prouver que la structure $\mathcal{A}$ satisfait la formule $\forall x \phi(x)$, on demande à un opposant à cette affirmation de choisir un élément quelconque du domaine et on doit être capable de prouver qu'il satisfait $\phi(x)$. On a ainsi donné, sous forme de jeu entre deux joueurs, la notion de satisfaction d'une formule dans une structure. On peut maintenant énoncer formellement le jeu d'Ehrenfeucht-Fraïssé.

**Définition 2.26** *Soient $\mathcal{A}$ et $\mathcal{B}$ deux $\tau$-structures. Le jeu $FO_m(\mathcal{A},\mathcal{B})$, pour $m \in \mathbb{N}$, est joué entre deux joueurs, $I$ et $II$. Chaque joueur doit jouer $m$ coups au cours de ce jeu. Au $i^{ème}$ coup, le joueur $I$ choisit l'une des deux structures $\mathcal{A}$ ou $\mathcal{B}$ et un élément de cette structure. Le joueur $II$ doit répliquer par un élément de l'autre structure. On illustre le choix des éléments après $m$ coups dans le tableau suivant (on ne s'intéressera pas à celui des joueurs $I$ ou $II$ qui a choisi la structure $\mathcal{A}$) :*

| $\mathcal{A}$ | $\mathcal{B}$ |
|---|---|
| $a_1$ | $b_1$ |
| $a_2$ | $b_2$ |
| $\vdots$ | $\vdots$ |
| $a_m$ | $b_m$ |

*Comme on le remarque dans ce tableau, les choix $\overline{a} = (a_1, \cdots, a_m)$ sont dans $\mathcal{A}$ et de même $\overline{b} = (b_1, \cdots, b_m)$ sont dans $\mathcal{B}$. Le joueur $II$* gagne *le jeu $FO_m(\mathcal{A},\mathcal{B})$ si, et seulement si, l'application $f : \overline{a} \to \overline{b}$ définie par :*

$$\forall i \in \{1, \cdots, m\},\ f(a_i) = b_i,$$

*est un isomorphisme entre $(\overline{a}, \tau^{\overline{a}})$ et $(\overline{b}, \tau^{\overline{b}})$. La structure $(\overline{a}, \tau^{\overline{a}})$ est la sous-structure de $\mathcal{A}$ dont le domaine est $\overline{a} \cup \overline{c}$, où $\overline{c}$ sont les éléments distingués de $\mathcal{A}$ qui interprètent les symboles de constantes dans $\tau$ ; de même pour $\overline{b}$ et $\mathcal{B}$.*

On remarque que, dans ce jeu, le joueur $II$ veut prouver que les structures sont semblables du point de vue la logique du premier ordre et le joueur $I$ est l'opposant.

**Définition 2.27** *On dit que le joueur $II$ a une* stratégie gagnante *pour le jeu $FO_m(\mathcal{A},\mathcal{B})$ s'il peut contrer tous les choix du joueur $I$, c'est-à-dire que pour tout $i \in \{1, \cdots, m\}$ il a un choix pour lequel il gagne le jeu $FO_i(\mathcal{A},\mathcal{B})$.*

Le jeu $FO_m(\mathcal{A},\mathcal{B})$ est *complet* dans le sens où si le joueur $II$ n'a pas de stratégie gagnante, alors le joueur $I$ en a une. Le résultat qui nous intéresse ici est le lien entre le fait qu'un des joueurs a une stratégie gagnante et le fait que les deux structures satisfont les mêmes formules jusqu'à un certain rang de quantification.

**Théorème 2.8 (Ehrenfeucht et Fraïssé)** *Soient $\mathcal{A}$ et $\mathcal{B}$ deux $\tau$-structures. Alors, les deux affirmations suivantes sont équivalentes :*
*– le joueur $II$ a une stratégie gagnante pour le jeu $FO_m(\mathcal{A},\mathcal{B})$ ;*

– *les structures $\mathcal{A}$ et $\mathcal{B}$ satisfont les mêmes énoncés du premier ordre de rang de quantification inférieur ou égal à $m$.*

Pour une preuve complète voir [EF95]. On va donner un exemple pour illustrer la méthode des choix des joueurs pour construire une stratégie gagnante.

**Exemple.** Soit $\tau$ une signature logique. Soient $\mathcal{A}$ et $\mathcal{B}$ deux $\tau$-structures finies. Supposons que $\mathcal{A}$ et $\mathcal{B}$ diffèrent sur une formule de rang de quantification 2. Par exemple, soit la formule $\psi \equiv \exists x \forall y \alpha(x, y)$, avec $\alpha$ formule ouverte, vérifiant :

$$\mathcal{A} \models \exists x \forall y \alpha(x, y) \, ;$$

et

$$\mathcal{B} \models \forall x \exists y \neg \alpha(x, y).$$

*Premier coup :* Le joueur $I$ choisit un élément $a_1 \in A$ tel que :

$$\mathcal{A} \models \forall y \alpha(a_1, y).$$

Un tel élément existe puisque :

$$\mathcal{A} \models \exists x \forall y \alpha(x, y).$$

Le joueur $II$ choisit un élément quelconque $b_1 \in B$.

*Second coup :* Le joueur $I$ choisit un élément $b_2 \in B$ tel que :

$$\mathcal{B} \models \neg \alpha(b_1, b_2).$$

Cet élément existe car :

$$\mathcal{B} \models \forall x \exists y \neg \alpha(x, y)$$

et en particulier pour $x = b_1$. Maintenant, le joueur $II$ peut choisir n'importe quel élément $a_2 \in A$, on aura toujours $\mathcal{A} \models \alpha(a_1, a_2)$ vu le choix qu'on a fait au premier coup. Ainsi $\{a_1, a_2\}$ et $\{b_1, b_2\}$ ne sont pas isomorphes, donc on a une stratégie gagnante pour le joueur $I$ dans le jeu $FO_2(\mathcal{A}, \mathcal{B})$. $\square Exemple$

Une conséquence de ce théorème est le corollaire suivant.

**Corollaire 2.3** *Soit $\mathcal{L}$ une classe de structures. Supposons que, pour tout $m \in \mathbb{N}$, il existe deux structures $\mathcal{A} \in \mathcal{L}$ et $\mathcal{B} \notin \mathcal{L}$ tel que le joueur $II$ a une stratégie gagnante pour le jeu $FO_m(\mathcal{A}, \mathcal{B})$. Alors $\mathcal{L}$ n'est pas définissable par un énoncé du premier ordre.*

**Exemple.** On va prouver que la classe des $\{<^2\}$-structures de taille paire n'est pas définissable au premier ordre, avec $<$ interprété comme un ordre strict. Pour ce fait on va prouver que pour tout $m > 0$, il existe deux structures, l'une de taille paire et l'autre de taille impaire, pour lesquelles le joueur $II$ a une stratégie gagnante.

Pour un $m > 0$ fixé, prenons les structure $S_1$ et $S_2$ représentant les mots $1 \cdots 1$, avec $2^m$ occurrences de 1, et $1 \cdots 1$, avec $2^m + 1$ occurrences de 1. La stratégie du joueur $II$ sera de répondre au $j^{\text{ème}}$ choix du joueur $I$ d'une façon à maintenir les conditions suivantes :

Pour tout $i < j$ on a :

1. Soit $|a_i - a_j|$ et $|b_i - b_j| > 2^{m-j}$ ;
2. Soit $|a_i - a_j| = |b_i - b_j| \leq 2^{m-j}$.

Pour une preuve détaillée, voir [EF95]. On a donc deux structures, l'une de taille paire et l'autre de taille impaire, qu'on ne peut distinger par aucune formule du premier ordre de rang de quantification $m$, pour tout $m$, donc cette propriété n'est pas définissable au premier ordre. □*Exemple*

### Cas de la logique monadique du second ordre

Pour étendre cette méthode à la logique du second ordre, on ne fait que rajouter des coups de choix de relations. Le joueur $I$ peut choisir une structure et une relation sur cette structure et le joueur $II$ doit répondre par une relation sur l'autre structure de même arité. On va expliquer le cas de la logique monadique du second ordre, la généralisation n'est guère plus compliquée en mettant les choix de relations de l'arité appropriée.

Rappelons que toute formule monadique du second ordre est équivalente à une formule monadique du second ordre sous forme prénexe.

**Définition 2.28** *Le jeu $MSO_{m,n}(\mathcal{A}, \mathcal{B})$ est le jeu joué comme suit :*

- *Pour chacun des $m$ premiers coups, le joueur $I$ choisit l'une des structures $\mathcal{A}$ ou $\mathcal{B}$ et un sous-ensemble de cette structure, le joueur $II$ répondant par un sous-ensemble de l'autre structure. Soient $S_1, \cdots, S_m$ les sous-ensembles choisis sur $\mathcal{A}$ et $R_1, \cdots, R_m$ les sous-ensembles choisis sur $\mathcal{B}$ (que ce soit par l'un ou l'autre joueur).*
- *Ensuite les deux joueurs jouent un jeu $FO_n(\mathcal{A}, S_1, \cdots, S_m; \mathcal{B}, R_1, \cdots, R_m)$.*

**Théorème 2.9** *Soient $\mathcal{A}$ et $\mathcal{B}$ deux $\tau$-structures finies. Alors, les deux affirmations suivantes sont équivalentes :*

- *le joueur $II$ a une stratégie gagnante pour le jeu $MSO_{m,n}(\mathcal{A}, \mathcal{B})$ ;*
- *les structures $\mathcal{A}$ et $\mathcal{B}$ satisfont les mêmes énoncés de la logique monadique du second ordre sous forme prénexe ayant $m$ quantificateurs du second ordre et $n$ quantificateurs du premier ordre.*

**Corollaire 2.4** *Soit $\mathcal{L}$ un langage. Supposons que, pour tout $m, n \in \mathbb{N}$, il existe deux structures $\mathcal{A} \in \mathcal{L}$ et $\mathcal{B} \notin \mathcal{L}$ telles que le joueur $II$ a une stratégie gagnante pour le jeu $MSO_{m,n}(\mathcal{A}, \mathcal{B})$. Alors $\mathcal{L}$ n'est pas définissable par un énoncé monadique du second ordre.*

Supposons maintenant qu'on veuille prouver qu'une propriété n'est pas définissable dans la partie existentielle de la logique $MSO$. Il nous suffit d'ajouter la contrainte que le joueur $I$, dans le jeu de $MSO$, choisit les ensembles dans la structure $\mathcal{A}$ et par conséquent le joueur $II$ choisit ses ensembles dans la structure $\mathcal{B}$. Pour les jeux de la logique existentielle, monadique du second ordre, voir les articles [Sch96, FSV95, Fag97], ou le livre de Ebbinghaus et Flum [EF95].

En étudiant ces jeux, Fagin a prouvé que $\exists MSO \neq \forall MSO$. Il a prouvé que la connexité des graphes finis n'était pas définissable dans $\exists MSO$, mais que, par contre, leur caractère non-connexe l'est. On reviendra sur ce résultat dans notre chapitre sur les jeux.

## 2.5 Complexité descriptive sur les mots

### 2.5.1 Introduction

On va donner, dans cette section, quelques définitions concernant la complexité descriptive. On parlera exclusivement d'un type particulier de structures finies à savoir *les structures de mots*, mais on donnera la méthode permettant de coder toute structure finie en une structure de mot.

**Remarque.** *Dans ce travail, on s'occupera exclusivement de structures de mots, sauf dans la section 2 du chapitre 6.* □

**Définition 2.29** *Soit $\Sigma$ un alphabet. On associe à $\Sigma$ la signature logique :*

$$\tau_\Sigma = (<, (P_a)_{a\in\Sigma}).$$

*On associe à chaque mot $w = w_1 \dots w_n$ sur $\Sigma$, la* structure de mot *$S_w$ qui est la structure relationnelle :*

$$S_w = ([n], <, (P_a)_{a\in\Sigma})$$

*où $[n]$ est le sous-ensemble $\{0, \dots, n-1\}$ de l'ensemble des entiers naturels $\omega$, où $<$ est l'ordre naturel sur $[n]$, et où $P_a$ est le prédicat unaire contenant les positions de $w$ étiquetées par $a$ :*

$$P_a = \{i \in [n] | w_i = a\}.$$

**Remarque.** Si on considère la classe des mots binaires, sur l'alphabet $\{0, 1\}$, on utilisera uniquement le prédicat contenant les positions étiquetées par 1, son complement contiendra donc les positions étiquetées par 0. □

**Exemple.** Soit le mot $w = aabbabb$ sur le vocabulaire $\Sigma = \{a, b\}$. La structure de mot qui lui est associée est :

$$S_w = (\{0, 1, 2, 3, 4, 5, 6\}, <, P_a^w = \{0, 1, 4\}, P_b^w = \{2, 3, 5, 6\}).$$

Soit le mot binaire $w = 1001101$. La structure qui lui est associée est :

$$S_w = (\{0, 1, 2, 3, 4, 5, 6\}, <, P^w = \{0, 3, 4, 6\}).$$

□*Exemple*

**Remarque.** Dans la suite, par abus de langage, on parlera de *logique* $\lambda$ pour désigner le langage de la logique $\lambda$ avec la sémantique appropriée. □

**Définition 2.30** *On dit qu'un langage $L$ sur un alphabet $\Sigma$ est défini par un énoncé $\phi$ d'une logique $\lambda$, sur la signature $\tau_\Sigma$ si, et seulement si,*

$$L = \{w \in \Sigma^* | S_w \models \phi\}.$$

**Exemple.** Soit $\phi$ la formule close :

$$\exists x \exists y (x > y \wedge P(x) \wedge P(y)).$$

Le langage défini par $\phi$ est l'ensemble des mots binaires qui comportent au moins deux 1. □*Exemple*

**Définition 2.31** *Un langage logique $\mathcal{L}$* capture, ou caractérise logiquement, *une classe de langages $\mathcal{C}$ si, pour tout langage $L \in \mathcal{C}$, il existe un énoncé $\phi \in \mathcal{L}$ tel que $\phi$ définit $L$ et, inversement, tout énoncé de $\mathcal{L}$ définit un langage de $\mathcal{C}$. On écrira, par abus de langage :*

$$\mathcal{C} = \mathcal{L}.$$

À titre d'exemple, le résultat de Fagin [Fag74] s'écrit comme suit :

$$\exists SO = NP.$$

On donnera d'autres exemples dans la section suivante.

**Codage :** On se restreint à l'étude des structures de mots dans ce travail car toute structure logique peut être codée par une structure de mot. Décrivons maintenant comment on peut coder une structure logique finie, sur une signature quelconque, par une structure de mot binaire.

Soit $\tau = (R_1^{ar_1}, \cdots, R_k^{ar_k}, c_1, \cdots, c_l)$ une signature quelconque. Soit la $\tau$-structure :

$$\mathcal{A} = (A, R_1^{\mathcal{A}}, \cdots, R_k^{\mathcal{A}}, c_1^{\mathcal{A}}, \cdots, c_l^{\mathcal{A}}),$$

avec $|A| = n$, où $|A|$ désigne la cardinalité de $A$. Soient $a_1, \cdots, a_n$ les éléments de $A$, dans un ordre quelconque. La relation $R_i^{\mathcal{A}}$ est un sous-ensemble de $A^{ar_i}$. On code cette relation en un mot binaire $bin^{\mathcal{A}}(R_i)$ de longueur $n^{ar_i}$. Chaque position correspond à un uplet, dans l'ordre lexicographique induit par l'ordre des éléments de $A$. Une position sera étiquetée 1 si, et seulement si, l'uplet correspondant est dans

$R_i^{\mathcal{A}}$. On code chaque constante $c_j^{\mathcal{A}}$ en son rang dans l'ordre de $A$ écrit en binaire. Le mot binaire correspondant à $\mathcal{A}$ est simplement la concaténation des mots binaires représentant ses relations et ses constantes :

$$bin_\tau(\mathcal{A}) = bin^{\mathcal{A}}(R_1) \cdots bin^{\mathcal{A}}(R_k) bin^{\mathcal{A}}(c_1) \cdots bin^{\mathcal{A}}(c_l).$$

On remarque que le mot binaire associé à la structure $\mathcal{A}$ a une taille polynomiale en la taille de $A$. On remarque qu'il existe $n!$ façons de coder chaque $\tau$-structure de taille $n$, une pour chaque ordre sur les éléments de la structure (voir le livre [Imm99] pour une discussion sur ce sujet).

**Hiérarchie :** Le pouvoir d'expression des logiques donne lieu, naturellement, à un ordre de complexité. Par exemple, si on peut définir un langage dans $FO$, on peut évidemment le définir dans $\exists MSO$ donc dans $MSO$ et dans $SO$. C'est pour cette raison que cette branche de la logique est appelée *complexité descriptive*. En effet, on classe les langages en fonction des ressources logiques qu'on utilise pour les décrire, comme le nombre de variables, le type des variables, le nombre de quantificateurs, etc.

### 2.5.2 Quelques résultats de complexité descriptive sur les mots

On a essayé de donner une caractérisation logique de chaque classe de complexité, du moins les plus célèbres, voir [Imm99, Pap94, Str94]. Cette description logique, indépendante du modèle de calcul sur lequel ces classes ont été défini, donne une plus grande légitimité et une certaine robustesse pour ces classes.

Le premier résultat dans cette direction est celui prouvé indépendamment par Büchi, Elgot et Trakhtenbrot [Büc60, Elg61, Tra61]. Il dit que la logique monadique du second ordre sur les structures de mots finis caractérise la classe des langages réguliers, autrement dit :

$$Reg = MSO.$$

Il a été prouvé que le fragment existentiel $\exists MSO$ de $MSO$, suffit pour caractériser $Reg$ (voir [Tho97, Pin96]) sur les structures de mots.

Ce résultat a conduit Eiter, Gottlob et Gurevich [EGG98] à trouver les classes préfixielles de $\exists SO$ équivalentes à $MSO$ sur les structures de mots. Ils ont prouvé que $\exists SO(\exists^*\forall\exists^* \cup \exists^*\forall^2)$ est incluse dans $Reg$ et qu'elle est maximale régulière. Grädel et Rosen [GR99] ont prouvé qu'en restreignant le nombre de variables dans la partie du premier ordre à 2, on obtient une nouvelle caractérisation logique de $Reg$. On écrit :

$$\exists SO(FO^2) = Reg.$$

Si on ne fait de restriction ni sur le préfixe, ni sur le nombre de variables dans la logique existentielle du second ordre, on caractérise la célèbre classe $NP$ des langages reconnus par une machine de Turing non-déterministe en temps polynomial. Ce résultat, dû à Fagin [Fag74], est vrai pour toute structure relationnelle, non seulement les structures de mots. Il s'écrit :

$$NP = \exists SO.$$

Stockmeyer [Sto76] a conclu que la logique du second ordre caractérise les langages de la hiérarchie polynomiale :

$$PH = SO.$$

Dans son article [Grä92], Grädel a caractérisé les classes $P$ et $NL$ des langages reconnaissables respectivement en temps polynomial sur une machine de Turing déterministe et en espace logarithmique par une machine de Turing non-déterministe. Ces caractérisations sont faites à partir de restrictions *syntaxiques* sur les formules

de $\exists SO$. Il a prouvé que :

$$P = \exists SO[\forall^*, Horn] \ ;$$

$$NL = \exists SO[\forall^*, 2 - clauses].$$

La restriction la plus naturelle de $SO$ est $FO$. McNaughton s'est intéressé au pouvoir d'expression de cette logique sur les structures de mots. Il a prouvé dans [MP71] que les langages définissables dans $FO$ correspondent exactement aux *langages sans étoile.* Les *langages sans étoile* sont les langages construits à partir des langages finis par opérations booléennes et concaténation. Wolfgang Thomas, voir [Tho97], a prouvé l'équivalence entre la hiérarchie *concaténation* et l'alternance de quantificateurs.

Une autre direction explorée est celle du pouvoir d'expression de (MSO) sur d'autres structures que les structures de mots finis. On donne quelques résultats dans le tableau qui suit.

| *Logique monadique du second ordre* | | |
|---|---|---|
| *Structures* | *Classe* | *Auteurs* |
| Mots infinis | Réguliers | [Büc60] |
| Arbres finis | Réguliers | [Don70, TW68] |
| Arbres infinis | Réguliers | [Rab69] |
| Graphes | ? ? ? | [Cou90] $\rightarrow$ |
| relationnelle | $MonadicNP$ | [FSV95] |

Pour plus de details sur le pouvoir d'expression de la logique monadique du second ordre, voir [Tho97, FSV95, Cou90].

| *Logique* | *Classe* | *Auteurs* |
|---|---|---|
| $(FO)$ | Sans étoile | [MP71, Pin96] |
| $\exists SO$ | $NP$ | [Fag74] |
| $SO$ | $PH$ | [Sto76] |
| $FO + DTC$ | $LogSpace$ | [Imm87] |
| $FO + TC$ | $NLogSpace$ | [Imm87] |
| $FO + LFP$ | $P$ | voir [Imm99] |
| $FO + PFP$ | $PSpace$ | voir [Imm99] |
| $\exists Match\ FO$ | (CFL) | [LST95] |

Pour plus de détails et de références, voir [Imm99, Str94]. On donnera dans cette thèse quelques résultats dans cette direction.

## 2.6 La logique $IMP$

La non validité de la propriété de Beth sur les structures finies a amené Kolaitis à s'intéresser en 1990 ([Kol90]) à la logique ajoutant les définitions implicites à la logique du premier ordre dans le cadre des modèles finis. Cette logique, appelée $IMP$, est l'ensemble des formules qui permettent de définir exactement les propriétés qui sont exprimables au premier ordre en utilisant des relations définies implicitement.

Intuitivement, cette logique est composée de formules du premier ordre sur une signature $\tau$ augmentée d'un ensemble de prédicats implicitement définissables. Elle correspond aussi à la sous-logique de la logique existentielle du second ordre dans laquelle toutes les variables du second ordre sont implicitement définissables dans le sens suivant :

**Définition 2.32** *Une formule $\phi(R)$ sur $\tau \cup \{R\}$ est une définition implicite sur la classe des $\tau$-structures $\mathcal{K}$, si toute $I \in \mathcal{K}$ admet une unique expansion en une $\tau \cup \{R\}$-structure satisfaisant $\phi(R)$.*

Cette définition est différente de la définition 2.23 de la page 31 dans le sens ou elle est totale. Essentiellement, la diffèrence entre ces deux définitions repose sur le fait que la definition 2.23 est équivalente à $\exists^{\leq 1} X \phi(X)$, et la définition 2.32 est équivalente à $\exists^{=1} X \phi(X)$, voir [EF95] page 213 pour plus de détails. Dans cette section, on utilisera la définition 2.32.

**Définition 2.33** *Une formule $\phi(\bar{x})$ de la logique $IMP$, où $\bar{x}$ est un n-uplet de variables du premier ordre, est la donnée d'un $(m+1)$-uplet de formules :*

$$(\psi_1(R_1), \ldots, \psi_m(R_m), \psi(\bar{x}, R_1, \ldots, R_m))$$

*tel que :*
- *les $\psi_i$ sont des formules closes du premier ordre sur $\tau \cup \{R_i\}$ ;*
- *$\psi$ est une formule du premier ordre sur $\tau \cup \{R_1, \ldots, R_m\}$ ;*
- *$\exists! R_1 \ldots \exists! R_m(\psi_1(R_1) \wedge \ldots \wedge \psi_m(R_m))$ est un énoncé vrai dans toutes les $\tau$-structures finies.*

**Définition 2.34** L'interprétation de la formule $\phi$ *ainsi définie est fixée par : sur chaque $\tau$-structure $\mathcal{A}$, on a :*

$$\mathcal{A} \models \forall X_1 \ldots X_m[(\psi_1(X_1) \wedge \ldots \wedge \psi_m(X_m)) \rightarrow \forall \bar{x}[\phi(\bar{x}) \leftrightarrow \psi(\bar{x}, X_1, \ldots, X_m)]].$$

**Exemple.** Reprenons l'exemple de la sous-section 2.4.2. Soit $\psi_1(P)$ la formule :

$$P(0) \wedge \forall x(P(x) \leftrightarrow \neg P(x+1)),$$

et soit $\psi(P,x)$ la formule :

$$P(max).$$

La formule $(\phi_1(P), \psi(P)) \in IMP$ définit la non parité de l'univers d'une structure logique représentant un segment initial de $\mathbb{N}$ muni de l'ordre induit, avec $max$ représentant le plus grand élément de ce segment. $\square Exemple$

**Définition 2.35** *Une machine de Turing* non ambiguë *est une machine de Turing non déterministe telle que, pour chaque mot accepté, il existe un unique calcul acceptant. Soit $UP$ (pour* unambiguous polynomial time*) la classe des problèmes reconnaissables en temps polynomial par un machine de Turing non ambiguë.*

Le premier résultat, dû à Kolaitis [Kol90], relie la logique $IMP$ au calcul sur des machines non-ambiguës.

**Résultat 2.3 (Kolaitis)** $UP \cap co-UP = IMP.$

**Définition 2.36** *Notons $Monadic-IMP$ la sous-logique de $IMP$ dans laquelle les seuls prédicats définis implicitement sont monadiques. Formellement, $Monadic-IMP$ est la classe de formules $\phi(\bar{x}) \in IMP$ de la forme :*

$$(\psi_1(R_1), \ldots, \psi_m(R_m), \psi(\bar{x}, R_1, \ldots, R_m)),$$

*dans laquelle tous les $R_i$ sont unaires.*

En partant du résultat de Büchi [Büc60], $\exists MSO = MSO = Reg$, et en remarquant que les notions de déterminisme, de non-déterminisme et de non-ambiguïté sur les automates finis définissent la même classe, qui est $Reg$, on peut déduire le résultat suivant.

**Théorème 2.10 ([Lac96])** *Un langage est régulier si, et seulement si, c'est l'ensemble des mots satisfaisant un énoncé de $Monadic-IMP$.*

**Preuve :**

La preuve est analogue à celle de Büchi, voir par exemple [Tho97]. Le changement essentiel est que l'automate, dont on essaye de mimer le calcul par une formule monadique du second ordre, est déterministe.

Le sens $Monadic - IMP \subseteq Reg$ est évident car la formule :

$$\exists R_1 \dots R_m(\psi_1(R_1) \wedge \dots \wedge \psi_m(R_m) \wedge \psi(\bar{x}, R_1, \dots, R_m))$$

définit le même langage que la formule :

$$(\psi_1(R_1), \dots, \psi_m(R_m), \psi(\bar{x}, R_1, \dots, R_m)),$$

puisqu'il existe $R_1, \cdots, R_m$.

Pour l'autre sens, on réécrit la preuve de Büchi pour un automate déterministe. L'idée est d'écrire un énoncé de $MSO$ qui exprime que l'automate accepte un mot. Soit $A = (Q, \Sigma, q_0, \delta, F)$ un automate fini déterministe. Sur un mot $w = w_0 \dots w_{n-1}$, l'énoncé établira l'existence d'un calcul $p_0 \dots p_n$ de $A$.

On commence par définir une variable unaire pour chaque état. Si $Q = \{0, \dots, k\}$, on considère les variables $X_0, \dots, X_k$. Intuitivement, la variable $X_i$ contient les positions de $w$ pour lesquelles $A$ est dans l'état $i$. Pour que le calcul soit acceptant, il faut et il suffit que les $X_i$ soient disjoints, que le premier état de l'automate soit 0 et que $p_n \in F$. La formule de $MSO$ est :

$$\exists X_0 \dots X_k(\textstyle\bigwedge_{i \neq j} \forall x \neg(X_i(x) \wedge X_j(x)) \wedge X_0(min) \wedge \forall xy(S(x,y) \rightarrow \\ \textstyle\bigvee_{j=\delta(i,a)}(X_i(x) \wedge P_a(x) \wedge X_j(y))) \wedge (\textstyle\bigvee_{\delta(i,a) \in F}(X_i(max) \wedge P_a(max))).$$

Il nous suffit maintenant de vérifier que cette formule est satisfaite par un unique $k+1$-uplet de sous-ensembles pour chaque mot. On voit que, si on commence par $X_0(min)$, les sous-ensembles sont inductivement construits suivant la fonction de transition, donc uniques. On conclut alors que $Reg = Monadic - IMP$. $\square Preuve$

## 2.7 Réseaux de Petri et logique

On va énoncer, dans cette section, le résultat de caractérisation logique des réseaux de Petri, donné par Parigot et Pelz dans [PP85]. Les auteurs ont introduit deux prédicats du second ordre, qu'ils ont noté $\leq_g$ et $=_g$, dont les interprétations sont des ordres partiels sur les sous-ensembles de $[n]$ définis par :

$$X \leq_g Y \text{ ssi pour tout } m \leq n, |X \cap [m]| \leq |Y \cap [m]|,$$
$$X =_g Y \text{ ssi } X \leq_g Y \text{ et } |X| = |Y|.$$

**Exemple.** Soit $\Sigma = \{(, )\}$. Un mot $w = u_0 \cdots u_{n-1}$ sur $\Sigma$ est une suite de parenthèses bien balancées (un mot du langage de Dyck à deux éléments) si, et seulement si :

$$S_w \models \{i < n | u_i = \text{`}')'\} =_g \{i < n | u_i = \text{`}('\}.$$

$\square Exemple$

**Définition 2.37** *Définissons* la logique monadique du second ordre avec cardinalité, *notée* $\mathbb{PP}$. *Soit* $\Sigma$ *un alphabet.*

1. *Les symboles de* $\mathbb{PP}(\Sigma)$ *sont :*
   - *des variables individuelles* $v_0, \ldots, v_n, \ldots$ ;
   - *des variables d'ensembles* $V_0, \ldots, V_n, \ldots$ ;
   - *les connecteurs* $\wedge, \vee, \neg, \rightarrow$ ;
   - *les quantificateurs* $\forall, \exists$ ;
   - *l'égalité* $=$ ;
   - *la relation du premier ordre* $<$ *(binaire) et* $P_a, a \in \Sigma$ *(unaire) ;*
   - *les symboles de relations du second ordre* $\leq_g$ *et* $=_g$ *(binaires).*
2. *Les formules atomiques de* $\mathbb{PP}(\Sigma)$ *sont du type* $x = y, x < y, Ux, U \leq_g V$, *et* $U =_g V$ *pour des variables individuelles* $x, y$ *et des variables d'ensembles ou des symboles de relations unaires* $U, V$.
3. *Les formules de* $\mathbb{PP}(\Sigma)$ *sont obtenues en appliquant les règles suivantes :*
   - *toute formule atomique est une formule ;*
   - *si* $\phi$ *et* $\psi$ *sont des formules, alors* $\phi \wedge \psi, \phi \vee \psi, \phi \rightarrow \psi, \neg\phi$ *sont des formules ;*
   - *si* $\phi$ *est une formule, et* $x$ *une variable individuelle, alors* $\forall x\phi$ *et* $\exists x\phi$ *sont des formules ;*
   - *si* $\phi$ *est une formule, et* $X$ *une variable d'ensemble, alors* $\forall X\phi$ *et* $\exists X\phi$ *sont des formules.*

4. *L'interprétation de* $=_g$ *et de* $\leq_g$ *se fait comme suit :*

$$X \leq_g Y \text{ ssi pour tout } m \leq n, |X \cap [m]| \leq |Y \cap [m]|,$$

$$X =_g Y \text{ ssi } X \leq_g Y \wedge |X| = |Y|.$$

**Résultat 2.4 (Parigot et Pelz [PP85])** *Soit L un langage sur un alphabet* $\Sigma$*. Les conditions suivantes sont équivalentes :*

1. *L est un langage de réseau de Petri ;*
2. *L est défini par un énoncé de* $\mathbb{PP}(\Sigma)$ *du type* $\exists\overline{X}\phi(\overline{X})$*, où* $\phi(\overline{X})$ *est une* $\mathbb{PP}(\Sigma)$*-formule du premier ordre ;*
3. *L est défini par un énoncé* $\mathbb{PP}(\Sigma)$ *du type* $\exists\overline{X}\phi(\overline{X})$*, où* $\phi(\overline{X})$ *est une combinaison positive*[3] *de formules du type* $X =_g Y$ *et des* $\mathbb{PP}(\Sigma)$*-formules du premier ordre dans lesquelles* $X =_g Y$ *et* $X \leq_g Y$ *ne figurent pas.*

**Preuve :**[Idée]

$1 \Rightarrow 2$ La formule va décrire les propriétés des suites franchissables et la partie invisible sera représentée par les ensembles quantifiés existentiellement qui représentent la production et la consommation du marquage.

$3 \Rightarrow 1$ Prouvé par induction sur la construction des formules.

$2 \Rightarrow 3$ Parigot et Pelz donnent les définitions de $\neg =_g, \leq_g, \neg \leq_g$ en utilisant uniquement le prédicat $=_g$ positivement.

Pour prouver la première implication, les auteurs ont donné une forme normale des réseaux de Petri. Ils ont appelé *réseau de Petri normal* tout réseau de Petri étiqueté sans $\varepsilon$ qui satisfait :

– Les fonctions d'incidence, arrière et avant, du réseau prennent leurs valeurs dans $\{0, 1\}$ au lieu de $\mathbb{N}$.
– Les marquages initial et finaux sont à valeurs dans $\{0, 1\}$ au lieu de $\mathbb{N}$.

Ils ont aussi prouvé que pour tout réseau de Petri $P$, il existe un réseau de Petri normal $P'$ tel que : $L(P) = L(P')$, voir [PP85] page 157. $\square Preuve$

[3]Cela veut dire qu'on n'utilise que les connecteurs $\wedge$ et $\vee$ dans la construction des formules.

## 2.8 Logique pour les langages algébriques

On va rappeler, dans cette section, un résultat de Lautemann et *al* [LST95] qu'on utilisera pour donner une caractérisation logique des langages algébriques non-ambigus dans le prochain chapitre.

**Définition 2.38** *Une relation binaire $M$ sur une structure de mot est dite un* appariement *(*matching *en anglais) si elle satisfait les propriétés suivantes :*

**1-** $\forall ij[(i,j) \in M \Rightarrow i < j]$ *;*

**2-** $\forall ij[(i,j) \in M \Rightarrow \forall k \neq i,j((i,k),\,(k,i),\,(j,k),(k,j) \notin M)]$ *;*

**3-** $\forall ijkl[(i,j),(k,l) \in M \Rightarrow (i < k < j \rightarrow i < l < j)]$.

*Soit $\psi_{Match}(M)$ la conjonction de ces trois conditions du premier ordre sur $\tau \cup \{M\}$. Soit* Match *la classe des appariements.*

On va rappeler la notion de restriction sémantique. On ne va donner la définition que pour le cas des appariements.

**Définition 2.39** *La logique $\exists Match\, FO$ est l'ensemble des formules $\exists Match\, \phi$, pour une formule $\phi \in FO$. Sémantiquement, $\exists Match \phi$ veut dire, pour une structure $S$, qu'il existe une relation $M \in Match$ telle que $(S, M) \models \phi$.*

**Exemple.** La formule de $\exists Match\, FO$ :

$$\exists M \forall x \forall y \exists z((M(x,z) \vee M(z,x)) \wedge (M(x,y) \rightarrow \bigvee_{1 \leq k \leq n} (P_{\alpha_k}(x) \wedge P_{\beta_k}(y))))$$

définit le langage de Dyck, voir par exemple [Har78], sur l'alphabet
$\Sigma_n = \{\alpha_1, \beta_1, \cdots, \alpha_n, \beta_n\}$. $\square Exemple$

Pour pouvoir définir un langage non régulier, on est obligé d'utiliser une logique plus puissante que $MSO$. D'un autre côté, la quantification existentielle d'un seul prédicat binaire exprime tous les langages algébriques et des langages $NP$-complets, selon un récent résultat d'Eiter, Gottlob et Gurevich [EGG98]. Dans [EGG98], les auteurs ont prouvé qu'une classe préfixielle de la logique existentielle du second ordre, ou bien n'exprime que des langages réguliers, ou exprime un problème $NP$-complet. De plus, ils ont prouvé que la $NP$-difficulté est présente par une formule de la forme $\exists R\phi$, pour un prédicat binaire $R$ et une formule $\phi$ du premier ordre du préfixe approprié.

Lautemann, Schwentick et Thérien [LST95] ont choisi une approche de restriction sémantique pour caractériser logiquement la classe des langages algébriques. Ils ont alors restreint le quantificateur du second ordre à être un appariement.

**Théorème 2.11 (Lautemann, Schwentick et Thérien [LST95])** *Un langage est algébrique si, et seulement si, il est la classe des modèles d'une formule de la forme $\exists M \in Match\ \phi$, où $\phi$ est une formule de la logique du premier ordre utilisant $M$.*

*Ce résultat reste vérifié si on remplace dans l'énoncé FO par MSO.*

Pour prouver ce résultat, Lautemann et *al* ont construit un énoncé du premier ordre sur $\tau \cup \{M\}$ pour chaque grammaire $G$, qui est vérifié par une structure de mot $S_w$ si, et seulement si, il existe un $G$-arbre de dérivation $T$ de $w$. De plus il existe une méthode effective de construire l'appariement à partir de l'arbre de dérivation.

Pour la seconde direction, ils ont combiné des résultats de Doner [Don70] et Thatcher et Wright [TW68], et Mezei et Wright [MW67], pour obtenir l'étape essentielle :

**Résultat 2.5** *Les trois propriétés suivantes sont équivalentes :*
- *$\mathcal{L}$ est un langage algébrique ;*
- *$\mathcal{L}$ est l'ensemble des mot d'un langage d'arbre régulier ;*
- *$\mathcal{L}$ est l'ensemble des mot d'un langage d'arbre qui satisfait un énoncé de la logique du second ordre monadique.*

Ensuite, ils ont construit des arbres à partir de chaque appariement sur chaque mot, et ont vérifié que cet ensemble d'arbres satisfait un énoncé de la logique monadique du second ordre.

En combinant ce résultat avec un résultat de Book et Greibach [BG70] qui dit :

*Un langage est dans $NTime[n]$ si, et seulement si, il est l'image par une projection de l'intersection d'un nombre fini de langages algébriques,*

Lautemann, Schwentick et Schweikardt [LSS99] ont énoncé le résultat suivant :

**Théorème 2.12 (Lautemann, Schweikardt et Schwentick [LSS99])**

$$NTime[n] = \exists M_1 \cdots M_k \exists \overline{R}(\phi_1 \wedge \cdots \wedge \phi_k) = \exists M_1 M_2 M_3 \exists \overline{R}(\phi_1 \wedge \phi_2 \wedge \phi_3)$$

*où les $M_i$ sont restreints à être des appariements, avec la condition que les seules relations binaires de $\phi_i$ sont $M_i$ et $<$, et tous les $R_j$ sont des variables unaires.*

**Preuve :**(de $\supseteq$ dûe à T. Schwentick [Got01])

Soit $L$ un langage défini par une formule du type :

$$\exists M_1 \cdots M_k \exists \overline{R}(\phi_1 \wedge \cdots \wedge \phi_k)$$

où les $M_i$ sont restreints à être des appariements, avec la condition que les seules relations binaires de $\phi_i$ sont $M_i$ et $<$, et tous les $R_j$ sont des variables unaires.

On commence par deviner les relations monadiques de $\bar{R}$ en temps non-deterministe linéaire. On considère que ces ensembles sont une augmantation de la signature. *On traite les $R_i$ comme des nouvelles lettres de l'alphabet, avec des modifications pour qu'une seule position n'ai pas deux étiquettes.*

Ensuite, on évalue les formules $\exists M_i \phi_i$, qui décrivent des langages algèbriques par le théorème 2.11, en temps non-deterministe linéaire. $\square Preuve$

Nous ne donnerons pas la preuve du second sens, car nous n'utilisons dans ce travail que le sens dont on a donné la preuve. En effet, on utilise ce résultat pour prouver qu'un langage est dans $NTime[n]$ en exhibant une formule de la forme donnée dans ce résultat.

Dans l'article [LMSV98], les auteurs ont donné une autre caractérisation logique des langages algèbriques, basée sur des quantificateurs généralisés de goupoïdes. Cette caractérisation utilise une caractérisation algébrique de cette classe qui dit que :"un langage est algébrique si, et seulement si, il peut être réduit à l'ensemble des solutions d'un problème de mots sur un goupoïde". On n'utilisera pas cette caractérisation dans cette thèse, pour plus de détails sur ce résultat voir [LMSV98].

## 2.9 La classe $RUD$.

La classe des *langages rudimentaires* a été introduite par Smullyan [Smu61], d'après une idée de Quine [Qui46], comme étant l'analogue, en théorie des langages, des relations *constructives en arithmétique,* au sens de Skolem. Les relations de l'*arithmétique constructive* sont définies à partir de l'addition et de la multiplication en utilisant les opérations booléennes, les transformations explicites et les quantifications bornées. La classe des *relations rudimentaires* est définie comme la plus petite classe de relations contenant la relation ternaire de concaténation ($Concat(x, y, z) \equiv xy = z$), les singletons $\{0\}$ et $\{1\}$, et close pour les opérations booléennes, *les transformations explicites* (ajout, renommage, permutation $\cdots$ de variables, voir [Smu61] pour une définition plus précise) et *les quantifications bornées* (les quantificateurs "sous-mot de" $\exists x \subseteq_p y, \forall x \subseteq_p y$ et les quantificateurs $\exists |x| \leq |y|, \forall |x| \leq |y|$, où $|x|$ est la longueur du mot $x$).

Dans sa thèse [Ben61], Bennett a prouvé que si on identifiait les entiers naturels à leurs représentations (dans une base de numération non unaire quelconque), les classes des relations rudimentaires et celles d'arithmétique constructive définissent le même concept, et de plus que ceci est indépendant de la base de numération ($m$-adique pour $m \geq 2$). Il a donc été prouvé qu'une relation sur les entiers est rudimentaire si, et seulement si, elle est définie par une formule de l'arithmétique constructives, *i.e.* une formule de la signature $\{+, \times\}$, dans laquelle tous les quantificateurs sont bornés par un polynôme en les variables libres.

Wrathall [Wra78] a prouvé que la classe des langages rudimentaires (les relations sur $\mathbb{N}$ peuvent être assimilées à des langages sur l'alphabet $\{0, 1\}$) est exactement la classe des langages de *la hiérarchie linéaire* $LinH$. Soit $LinTime$ (resp. $NLinTime$) la classe des langages reconnaissables en temps linéaire par une machine de Turing déterministe (resp. non-déterministe). La hiérarchie linéaire, $LinH$, est définie inductivement par :

$$\Sigma_0^{lin} = LinTime \text{ et } \Sigma_{i+1}^{lin} = NLinTime^{\Sigma_i^{lin}},$$

$$LinH = \cup_{i \in \mathbb{N}} \Sigma_i^{lin}.$$

**Théorème 2.13 ([Ben61, Wra78])** *La classe des relations rudimentaires, celle des langages définis par des formules de l'arithmétique bornée et celle des langages de la hiérarchie linéaire coïncident.*

On peut situer $LinH$ par rapport aux autres classes de complexité grâce aux résulats :

$$LogSpace \subseteq LinH \subseteq DLinSpace,$$

prouvés par Nepomnjascii, voir par exemple [HP93], pour la première inclusion, et Myhill [Myh60] pour la seconde.

Un grand nombre de problèmes de complexité structurelle sont liés à des questions sur les langages rudimentaires. L'un des plus célèbres est celui énoncé par Woods [Woo81] qui dit que si $RUD$ est égal aux spectres binaires alors $NP \neq co - NP$. Un autre problème ouvert, qui a des conséquences en théorie de la complexité algorithmique, est le problème de comptage, voir [PW86, DLM98] ; on reviendra ultérieurement sur ce problème.

De plus cette classe est liée à des schémas restreints de récurrence. Pour plus de détails voir l'article d'Henri-Alex Esbelin et Malika More [EM98]. Suite à la convergence de toutes ces définitions, la classe des langages rudimentaires apparaît comme une classe naturelle, et surtout robuste. De plus cette classe admet une caractérisation logique très naturelle. Le résultat, dû à Lynch et Immerman [Lyn82, Imm87], dit qu'un langage est rudimentaire ssi il est la classe des modèles d'un énoncé de la logique monadique du second ordre avec l'addition. On rappelle les résultats de Lynch et Immerman.

**Résultat 2.6 ([Lyn82])** *Pour tout $k \in \mathbb{N}^*$,*

$$NTime[n^k] \subseteq \exists SO(+, arity\ k).$$

*$\exists SO(+, arity\ k)$ est le fragment de la logique existentielle du second ordre avec l'addition dans lequel les variables du second ordre sont d'arité au plus $k$.*

Soit $PH - Time[f(n)]$ la classe des langages reconnus par une machine de Turing alternante en temps $f(n)$, en alternant un nombre fini de fois entre les états existentiels et les états universels, voir par exemple [Pap94, Imm99] pour une définition plus détaillée. Bien que la réciproque du résultat de Lynch soit toujours ouverte, Immerman a prouvé qu'en passant à l'alternance des appels, on atteint l'équivalence. Ceci est dû à la robustesse de l'alternance.

**Résultat 2.7 ([Imm87])** *Avec les mêmes notations que le résultat de Lynch, on a :*

$$SO(+, arity\ k) \equiv PH - Time[n^k].$$

Si on se restreint à l'arité 1 (la logique monadique du second ordre), on obtient une caractérisation de $LinH$ et donc, en utilisant le résultat de Wrathall [Wra78], on a une caractérisation logique de $RUD$. Dans [MO97], Frédéric Olive et Malika More ont donné une preuve directe de ce résultat, sans passer par le résultat de Wrathall.

On peut renvoyer à l'article [EM98] pour plus de détails et de références à propos de cette classe.

# Chapitre 3

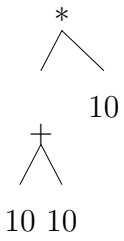
# Une caractérisation logique de la classe des langages algébriques non-ambigus

**Résumé**

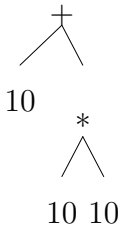
On donne, dans une première partie, une caractérisation logique de la classe des langages algébriques non-ambigus (THÉORÈME 1), en utilisant une caractérisation logique des langages algébriques de [LST95]. On en déduit, dans une deuxième partie, que la logique d'une relation implicite binaire est indécidable (THÉORÈME 2). On donne ensuite une nouvelle preuve du résultat de Gurevich selon lequel il existe des propriétés implicitement définissables qui ne sont pas explicitement définissables.

## 3.1 Introduction et énoncé du premier théorème

La notion d'ambiguïté est présente dans tous les langages. En langage naturel, une *phrase* est dite *ambiguë* si deux personnes différentes peuvent l'interpréter de deux façon différentes. Un langage de programmation doit être non-ambigu dans le sens qu'il ne doit pas exister *des arbres de dérivations* différents pour un même programme. En effet, sinon il existe plusieurs manières d'analyser ce programme, avec en général des résultats attendus différents. Prenons, par exemple, l'évaluation de l'expression arithmétique $10 + 10 * 10$. Selon l'arbre de dérivation retenu, voir les figures *Arbre 1* et *Arbre 2* suivantes, cette expression aura la valeur 110 ou 200.




*Arbre 1.*




*Arbre2.*

La définition formelle de l'ambiguïté des langages algébriques se fait grâce à la notion d'*arbre de dérivation,* généralisant l'exemple que nous venons de donner.

Dans le cas des automates à piles, les notions de déterminisme, de non-déterminisme et de non-ambiguïté sont strictement diffèrentes, comme le montrent les résultats suivants :

$$DCFL \subsetneq UCFL \subsetneq CFL,$$

où DCFL désigne la classe des langages algébriques déterministes (pour l'anglais Deterministic Context Free Languages). Les preuves complètes sont données dans [Har78].

On va donner, dans ce chapitre, une caractérisation logique des langages algébriques non-ambigus. Cette caractérisation s'appuie sur un résultat de Lautemann, Schwentick et Thérien [LST95] et sur la notion de définitions implicites (définition 2.23). On va prouver que la classe des langages algébriques non-ambigus correspond à la classe des langages définis par une formule dans laquelle un prédicat binaire est définit implicitement et est de plus restreint à être un appariement.

**THÉORÈME 1** *Un langage est algébrique non-ambigu si, et seulement si, il est définissable dans* $(IMP_2)^{Match}$.

## 3.2 Définition de $IMP_2^{Match}$

**Définition 3.1** *Soit $IMP_2^{Match}$ la logique qui utilise la définition implicite, dans le sens de la définition 2.23 page 31, d'un unique prédicat, qui est astreint à être un appariement.*

**Remarque.** On peut donner une définition de cette logique comme sous-logique de $\exists Match\ FO$ en restreignant l'appariement à être implicitement définissable, donc, à être unique ; i.e. de la forme suivante :

$$\exists! M \in Match\ \phi,$$

avec $\phi$ une formule du premier ordre sur $\tau \cup \{M\}$. □

**Exemple.** La formule de $\exists Match\ FO$ :

$$\exists M \forall x \forall y \exists z((M(x,z) \vee M(z,x)) \wedge (M(x,y) \rightarrow \bigvee_{1 \leq k \leq n} (P_{\alpha_k}(x) \wedge P_{\beta_k}(y))))$$

définit le langage de Dyck, voir par exemple [Har78], sur l'alphabet :

$$\Sigma_n = \{\alpha_1, \beta_1, \cdots, \alpha_n, \beta_n\}.$$

Cette formule est dans $(IMP_2)^{Match}$ car l'appariement qui satisfait cette formule est unique, puisque à chaque parenthèse ouvrante correspond une unique parenthèse fermante avec la condition que le mot entre ces deux positions aussi soit dans le langage de Dyck. □*Exemple*

## 3.3 Démonstration de la condition $(UCFL \subseteq (IMP_2)^{Match})$

Pour prouver cette direction, on va suivre la preuve de Lautemann et *al* [LST95], qui dit qu'un langage est algébrique si, et seulement si, il est la classe des modèles d'une formule de $\exists Match\ FO$, et vérifier attentivement qu'à chaque étape de la preuve on peut préserver le caractère non-ambigu de notre grammaire.

### 3.3.1 Première étape : *Construction de la grammaire appropriée.*

Une des étapes clé de la preuve de cette direction est le raffinement du lemme suivant.

**Lemme 3.1 (Forme normale double de Greibach, [Hot78])** *Chaque langage algébrique non-ambigu admet une grammaire algébrique non-ambiguë*

$$G = (\Sigma, N, S, P),$$

*dont toutes les productions sont de l'une des deux formes suivantes :*

1. $X \to a$, *pour* $X \in N$ *et* $a \in \Sigma$ ;
2. $X \to aub$, *avec* $X \in N$, $a, b \in \Sigma$, *et* $u \in (\Sigma \cup N)^*$.

*Une telle grammaire est dite sous* forme normale double de Greibach.

Dans leur preuve [LST95], les auteurs ont utilisé une forme normale plus fine que la forme normale double de Greibach, à savoir une forme dans laquelle on peut deviner le membre gauche d'une production en connaissant uniquement les terminaux de son membre droit. On va mimer la preuve de ce résultat dans la preuve du lemme suivant dans le cas des langages non-ambigus.

**Définition 3.2** *Soit* $p : X_0 \to v_0 X_1 v_1 \dots v_{s-1} X_s v_s$ *une production de la grammaire*

$$G = (\Sigma, N, S, P),$$

*où* $X_0 \dots X_s \in N$ *et* $v_0 \dots v_s \in \Sigma^*$.

*Le* motif *de la production* $p$ *est le mot, noté* $motif(p)$, *sur l'alphabet* $\Sigma \cup \{|\}$, *où* $|$ *est un symbole supplémentaire ne figurant pas parmi les symboles de* $\Sigma \cup N$, *défini par :*

$$motif(p) = v_0|v_1|\dots|v_{s-1}|v_s.$$

Le lecteur peut, s'il le désire, suivre l'illustration des résultats et des méthodes de cette partie de la preuve sur l'exemple de la section 3.3.4.

**Lemme 3.2** *Chaque langage algébrique non-ambigu admet une grammaire algébrique non-ambiguë $G = (\Sigma, N, S, P)$, en forme normale double de Greibach, telle que si deux productions ont le même motif alors elles ont le même membre gauche.*

**Preuve :** *du lemme 3.2.*

Le processus est le même que celui donné dans [LST95] dans le cas des grammaires algébriques. Soit $L$ un langage algébrique non ambigu. Soit $G$ une grammaire sous forme normale double de Greibach définissant $L$, qui existe d'après le lemme 3.1.

*Étape 1.1. On élimine toutes les productions de $G$ de la forme $X \to \alpha$, avec $\alpha \in \Sigma$ et $X \in N - \{S\}$, en introduisant à la place, une production $Y \to u\alpha v$ pour chaque production $Y \to uXv \in P$.*

**Exemple.** Soit la grammaire $G = (\Sigma, N, S, P)$ telle que :

$$P = \{S \to aABa, B \to aAbb, A \to c\} \cup P',$$

où, dans $P'$, le non terminal $A$ ne figure dans le membre droit d'aucune production. Après le procédé décrit plus haut, on obtient le nouvel ensemble de productions :

$$P'' = \{S \to aABa, S \to acBa, B \to aAbb, B \to acbb\} \cup P'.$$

On peut vérifier aisement que si on peut dériver un mot $w$ d'une façon unique par $P$, on ne peut le dériver que d'une façon unique par $P''$. $\square Exemple$

**Remarque :** Cette élimination augmente considérablement le nombre de productions, mais on peut facilement prouver qu'on obtient une grammaire équivalente, et que l'ambiguïté de la nouvelle grammaire impliquerait celle de la grammaire de départ.

*Étape 1.2. On énumère tous les non-terminaux $X_1, ..., X_N$ d'une façon arbitraire, avec la seule restriction $X_1 = S$. Pour $i = 2$ à $N$, on fait ce qui suit pour chaque $i$ : tant qu'il y a une production $p = X_i \to v$ dont le motif apparaît aussi comme motif d'une production avec un membre gauche $X_j$, pour un $j < i$, on remplace $p$ par toutes les productions qui peuvent être obtenues à partir de $p$ par substitution d'un des non-terminaux de $v$ par toutes les possibilités.*

Le fait de remplacer ces productions par les productions appropriées et la non-ambiguïté de la grammaire de départ, impliquent qu'on obtient ainsi une grammaire équivalente et qu'elle est non-ambiguë. Il suffit de remarquer qu'à chaque fois qu'on introduit une production, on élimine celle qui peut rendre la grammaire ambiguë.

On peut vérifier que la grammaire résultante satisfait la propriété : "deux productions ont le même motif si, et seulement si, il ont le même membre gauche".

□*Preuve du lemme 3.2*

### 3.3.2 Seconde étape : *Construction de l'appariement associé à la grammaire.*

Soit $L$ un langage algébrique non-ambigu. Soit :

$$G = (N, \Sigma, S, P)$$

une grammaire non-ambiguë, satisfaisant les conditions du lemme 3.2, engendrant $L$. Soit un mot $w \in L$ . Soit $T$ l'arbre de dérivation de $w$ (unique d'après la non-ambiguïté de la grammaire). Le fils le plus à gauche, et le fils le plus à droite, de chaque nœud interne de $T$ est une feuille, étiquetée par un symbole terminal grâce à la forme normale double de Greibach.

*On construit* l'appariement $M_T$ associé à l'arbre $T$ *comme suit :*

> $(i, j) \in M_T$ *si, et seulement si, i correspond au fils le plus à gauche, et j au fils le plus à droite du même nœud interne de* $T$.

**Exemple.** Soit $T$ l'arbre de dérivation du mot $w = abbbbaab$ sur $\Sigma = \{a, b\}$ de la figure 3.1. Les arcs représentés sur la figure représentent l'appariement. L'appariement défini sur les positions de :

$$S_w = (\{1, 2, 3, 4, 5, 6, 7, 8\}, <, P_a = \{1, 6, 7\}, P_b = \{2, 3, 4, 5, 8\})$$

est :

$$M_T = \{(1, 8),\ (2, 5),\ (3, 4),\ (6, 7)\}.$$

□*Exemple*

### 3.3.3 Troisième étape : *Construction de la formule associée.*

Lautemann, Schwentick et Thérien ont prouvé, dans [LST95], qu'il existe une formule $\phi$ sur $(\Sigma, <, M)$ qui est vérifiée par un mot $w$ et un appariement $M$ si, et seulement

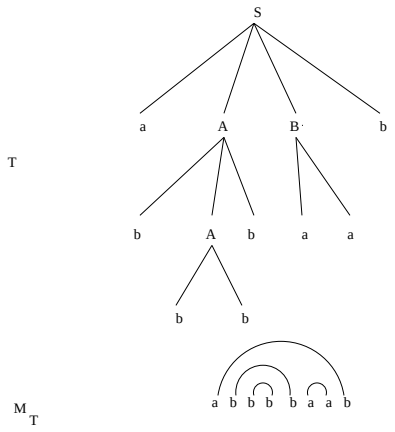



FIG. 3.1 – Un arbre de dérivation, $T$, de $w = abbbbaab$ et l'appariement $M_T$.

si, il existe un $G-$arbre de dérivation $T$ de $w$ tel que $M = M_T$. Il en découle qu'il existe un unique appariement $M$ sur $w$ avec $(w, M) \models \phi$ si, et seulement si, $w$ peut être dérivé dans $G$. La suite de la preuve de cette direction est consacrée à la construction de cette formule et à la vérification qu'il s'agit d'une définition implicite de $M$.

**Définition 3.3** *1. Chaque arc $(i, j)$ de $M$ définit un sous-mot $w_i \cdots w_j$ de $w$.*

2. *On dit qu'un arc $(k, l) \in M$ $(i < k < l < j)$* se trouve sur la surface de *$(i, j)$ s'il n'existe aucun arc entre lui et $(i, j)$.*

3. *Une position $k$ qui n'est pas le sommet d'un arc autre que $(i, j)$ se trouve sur la suface de $(i, j)$ s'il n'y a aucun arc entre elle et $(i, j)$.*

4. *Le mot formé des symboles sur la surface de $(i, j)$, en remplaçant les arcs de la surface par $|$, est appelé le* motif *de $(i, j)$.*

**Exemple.**(Continué) Si on reprend l'exemple de la figure 3.1, le motif de $(2, 5)$ est $b|b$. $\square Exemple$

On dit qu'un *arc $(i, j)$ correspond à une production $p$* si le motif de l'arc et celui de la production sont identiques. Cette propriété est facilement exprimée au premier ordre. Soit $\chi_p(i, j)$ la formule disant que l'arc $(i, j)$ correspond à la production $p$.

Il est facile de construire, pour chaque mot $v$, une formule $\psi_v(x, y)$ disant que le sous-mot, de l'univers de la structure, strictement compris entre les positions $x$ et $y$ est $v$.

Pour $p \equiv \alpha v_0 X_1 v_1 \cdots v_{s-1} X_s v_s \beta$, on construit la formule $\chi_p(x, y)$, disant que, entre les positions $x$ et $y$, le motif de l'arc est identique au motif de la production $p$, de la façon suivante :

$$P_\alpha(x) \wedge P_\beta(y) \wedge \exists x_1 y_1 \cdots x_s y_s (x < x_1 < y_1 < \cdots < x_s < y_s < y \wedge$$
$$\wedge \psi_{v_0}(x, x_1) \wedge \psi_{v_1}(y_1, x_2) \wedge \cdots \wedge \psi_{v_s}(y_s, y) \wedge M(x_1, y_1) \wedge \cdots \wedge M(x_s, y_s)).$$

Pour chaque $X \in N$, soit $\bar{\chi}_X(x, y)$ la disjonction de tous les $\chi_p(x, y)$ pour lesquelles $X$ est le membre gauche de la production $p$. Pour $p \equiv \alpha v_0 X_1 v_1 \cdots v_{s-1} X_s v_s \beta$, la formule $\bar{\chi}_p(x, y)$ exprime que l'arc $(x, y)$ correspond à la production $p$ et que les arcs

$$(x_1, y_1), \cdots, (x_s, y_s)$$

situés sur la surface de $(x, y)$ correspondent respectivement à des productions dont les membres gauches sont $X_1, \cdots, X_s$.

On a alors :

$$\bar{\chi}_p(x, y) \equiv \chi_p(x, y) \wedge \bar{\chi}_{X_1}(x_1, y_1) \wedge \cdots \wedge \bar{\chi}_{X_s}(x_s, y_s).$$

En conclusion, la formule $\phi$ recherchée, qui définit le langage $L$ est :

$$\bigvee_{S \to u \in P} \psi_u(min, max) \vee [M(min, max) \wedge \bar{\chi}_S(min, max) \wedge \forall xy (M(x, y) \to \bigvee_{p \in P} \bar{\chi}_p(x, y))].$$

Puisque chaque production est uniquement déterminée par son motif, on a une formule $\phi$ qui définit l'appariement $M$ implicitement.

### 3.3.4 Exemple

Dans cette section on va construire la formule correspondant à un langage non ambigu par la méthode décrite plus haut. Ceci illustre la preuve de cette partie du théorème.

Soit $\Sigma = \{a, b\}$.

Soit :

$$L = \{a\} \cup \{a^{2n} b^{2m}; n, m > 0\} \cup \{a b^{2n+1} a; n \geq 0\}$$

un langage sur l'alphabet $\Sigma$.

Soit la grammaire sous forme normale double de Greibach :

$$G = (\Sigma, N = \{S, A, B\}, S, P)$$

telle que $P =$ :

$$\begin{array}{lll} S & \rightarrow & a; aABb; aBa\,; \\ A & \rightarrow & a; aAa\,; \\ B & \rightarrow & b; bBb. \end{array}$$

On peut facilement vérifier que cette grammaire engendre bien le langage $L$, et est non-ambigüe.

*Étape 1.1.*

Les productions qui nous intéréssent sont $p_1 = A \rightarrow a$ et $p_2 = B \rightarrow b$ (La production $S \rightarrow a$ n'est pas à éliminer).

Pour éliminer $p_1$, on cherche les productions qui ont $A$ dans leurs membre droit, à savoir $p_3 = S \rightarrow aABb$ et $p_4 = A \rightarrow aAa$. On obtient une nouvelle grammaire dont l'ensemble de productions est $P' =$ :

$$\begin{array}{lll} S & \rightarrow & a; aABb; aaBb; aBa\,; \\ A & \rightarrow & aaa; aAa\,; \\ B & \rightarrow & b; bBb. \end{array}$$

Pour éliminer $p_2$, on procède de la même manière en traitant les productions $S \rightarrow aABb; aaBb; aBa$ et $B \rightarrow bBb$. On obtient la nouvelle grammaire dont l'ensemble des productions est $P'' =$ :

$$\begin{array}{lll} S & \rightarrow & a; aABb; aaBb; aabb; aAbb; aba; aBa\,; \\ A & \rightarrow & aaa; aAa\,; \\ B & \rightarrow & bbb; bBb. \end{array}$$

*Étape 1.2.*

Considèrons la numération des non-terminaux telle que $S = X_1,\ A = X_2$ et $B = X_3$. les motifs des productions ayant $S$ comme membre gauche sont :

$$\{a, a||b, aa|b, aabb, a|bb, aba, a|a\}.$$

On peut remarquer que la production $A \rightarrow aAa$ a le même motif que la production $S \rightarrow aBa$. On remplace alors la production $A \rightarrow aAa$ par les productions $A \rightarrow aaaaa; aaAaa$. On obtient donc la grammaire dont les productions sont :

$$\begin{array}{lll} S & \rightarrow & a; aABb; aaBb; aabb; aAbb; aba; aBa\,; \\ A & \rightarrow & aaa; aaaaa; aaAaa\,; \\ B & \rightarrow & bbb; bBb. \end{array}$$

Cette grammaire satisfait bien les conditions du lemme 3.2.

*Étapes 2+3.*

$p_1 = S \to a\,;$
$p_2 = S \to aba\,;$
$p_3 = S \to aabb\,;$
$p_4 = S \to aAbb\,;$
$p_5 = S \to aaBb\,;$
$p_6 = S \to aABb\,;$
$p_7 = S \to aBa\,;$
$p_8 = A \to aaa\,;$
$p_9 = A \to aaaaa\,;$
$p_{10} = A \to aaAaa\,;$
$p_{11} = B \to bbb\,;$
$p_{12} = B \to bBb.$

Les formules associée à ces productions sont respectivement :

$\chi_1(i,j) = (i = j \wedge P_a(i))$
$\chi_2(i,j) = \exists x(P_a(i) \wedge P_b(x) \wedge P_a(j) \wedge succ(i,x) \wedge succ(x,j)$
$\chi_3(i,j) = \exists xy(P_a(i) \wedge p_a(x) \wedge P_b(y) \wedge P_b(j) \wedge succ(i,x) \wedge succ(x,y) \wedge succ(y,j))$
$\chi_4(i,j) = \exists xyz(P_a(i) \wedge M(x,y) \wedge P_b(z) \wedge P_b(j) \wedge succ(i,x) \wedge succ(y,z) \wedge succ(z,j))$
$\chi_5(i,j) = \exists xyz(P_a(i) \wedge P_a(x) \wedge M(y,z) \wedge P_b(j) \wedge succ(i,x) \wedge succ(x,y) \wedge succ(z,j))$
$\chi_6(i,j) = \exists xyzt(P_a(i) \wedge M(x,y) \wedge M(z,t) \wedge P_b(j) \wedge succ(i,x) \wedge succ(y,z) \wedge succ(t,j))$
$\chi_7(i,j) = \exists xy(P_a(i) \wedge M(x,y) \wedge P_a(j) \wedge succ(i,x) \wedge succ(y,j))$
$\chi_8(i,j) = \exists x(P_a(i) \wedge P_a(j) \wedge P_a(x) \wedge succ(i,x) \wedge succ(x,j))$
$\chi_9(i,j) = \exists xyz(P_a(i) \wedge P_a(j) \wedge P_a(x) \wedge P_a(y) \wedge P_a(z) \wedge succ(i,x) \wedge succ(z,j)) \wedge succ(x,y) \wedge succ(y,z))$
$\chi_{10}(i,j) = \exists xyzt(P_a(i) \wedge P_a(j) \wedge P_a(x) \wedge P_a(t) \wedge M(y,z) \wedge succ(i,x) \wedge succ(x,y) \wedge succ(z,t) \wedge succ(t,j))$
$\chi_{11}(i,j) = \exists x(P_b(i) \wedge P_b(j) \wedge P_b(x) \wedge succ(i,x) \wedge succ(x,j))$
$\chi_{12}(i,j) = \exists xy(P_b(i) \wedge M(x,y) \wedge P_b(j) \wedge succ(i,x) \wedge succ(y,j))$

On obtient alors :

$$\bar{\chi}_X = \chi_4 \vee \cdots \vee \chi_7$$

$$\bar{\chi}_A = \chi_8 \vee \chi_9 \vee \chi_{10}$$

$$\bar{\chi}_B = \chi_{11} \vee \chi_{12}$$

$\bar{\chi}_4(i,j) = \exists xyz(P_a(i) \wedge M(x,y) \wedge P_b(z) \wedge P_b(j) \wedge succ(i,x) \wedge succ(y,z) \wedge succ(z,j) \wedge \bar{\chi}_A(x,y))$

$\bar{\chi}_5(i,j) = \exists xyz(P_a(i) \wedge P_a(x) \wedge M(y,z) \wedge P_b(j) \wedge succ(i,x) \wedge succ(x,y) \wedge succ(z,j) \wedge \bar{\chi}_B(y,z))$
$\bar{\chi}_6(i,j) = \exists xyzt(P_a(i) \wedge M(x,y) \wedge M(z,t) \wedge P_b(j) \wedge succ(i,x) \wedge succ(y,z) \wedge succ(t,j) \wedge \bar{\chi}_A(x,y) \wedge \bar{\chi}_B(z,t))$
$\bar{\chi}_7(i,j) = \exists xy(P_a(i) \wedge M(x,y) \wedge P_a(j) \wedge succ(i,x) \wedge succ(y,j) \wedge \bar{\chi}_B(x,y))$
$\bar{\chi}_{10}(i,j) = \exists xyzt(P_a(i) \wedge P_a(j) \wedge P_a(x) \wedge P_a(t) \wedge M(y,z) \wedge succ(i,x) \wedge succ(x,y) \wedge succ(z,t) \wedge succ(t,j) \wedge \bar{\chi}_A(y,z))$
$\bar{\chi}_{12}(i,j) = \exists xy(P_b(i) \wedge M(x,y) \wedge P_b(j) \wedge succ(i,x) \wedge succ(y,j) \wedge \bar{\chi}_B(x,y))$

Pour les autres productions, les formules $\bar{\chi}_\alpha(i,j)$ sont égales aux formules $\chi_\alpha(i,j)$.

La formule qui caractérise le langage $L$ est :

$\chi_1(min,max) \vee \chi_2(min,max) \vee \chi_3(min,max) \vee$
$[\forall xy(M(x,y) \rightarrow \bigvee_{p \in P} \bar{\chi}_p(x,y)) \wedge (M(min,max) \wedge \bar{\chi}_S(min,max))]$.

## 3.4 Démonstration de la condition $(UCFL \supseteq (IMP_2)^{Match})$

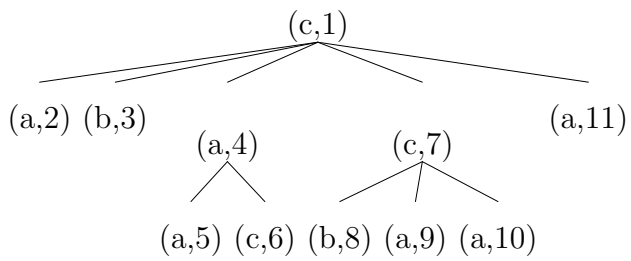
**Notations :** Commençons par fixer quelques notations sur les arbres.

- Les arbres que nous utilisons sont des arbres étiquetés par un ensemble fini d'étiquettes, et ordonnés dans un ordre en profondeur avec priorité à gauche, ce qui correspond à l'ordre préfixé. L'ensemble $\Sigma$ des étiquettes d'arité 0 est dit *l'alphabet de feuilles.*
- Les étiquettes de feuilles d'un arbre, concaténées suivant la relation d'ordre (en profondeur avec priorité à gauche), forment un mot de $\Sigma^*$, appelé *le mot de* $T$.
- Un arbre peut être assimilé à une structure logique relationnelle, voir [Tho97]. Si on définit l'univers d'un arbre comme l'ensemble de ses nœuds, on définit une relation unaire $P_a$, pour chaque étiquette $a$, et une relation binaire $C$, telle que $(i,j) \in C \longleftrightarrow i$ est un fils de $j$, et, finalement la relation d'ordre $<$ sur l'ensemble des nœuds de l'arbre.

  Dans ce langage logique, on peut facilement exprimer les propriétés suivantes dans $MSO$ :
  - $Lf(i) : i$ est une feuille ;
  - $Lc(i,j) : i$ le fils le plus à gauche de $j$ ;
  - $Rc(i,j) : i$ le fils le plus à droite de $j$ ;
  - $An(i,j) : i$ est un ancêtre de $j$ ;
  - $Pt(U,i,j) : An(i,j)$, $Lf(i)$ et $U$ est l'ensemble contenant exactement les nœuds sur le chemin entre $i$ et $j$.
- Soit T un arbre et $w$ un mot, on dit que $w$ est le mot de $T$, et on note $w = Yield(T)$ si, et seulement si, le mot constitué des feuilles de $T$ concaténées dans l'ordre (de gauche à droite) est $w$.

**Exemple.**

Soit l'arbre étiqueté dans l'alphabet $\{a,b,c\}$ (à titre d'illustration, on a étiqueté les positions de cet arbre par leurs étiquettes et leurs rangs) :



La structure logique correspondant à cet arbre admet comme univers :

$$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\},$$

avec l'ordre usuel sur ce segment initial de $\mathbb{N}$, les relations unaires :

$$P_a = \{2, 4, 5, 9, 10, 11\},$$

$$P_b = \{3, 8\},$$

$$P_c = \{1, 6, 7\},$$

et la relation $C$ est formée des couples :

$$\{(1,2), (1,3), (1,4)(1,7), (1,11), (4,5), (4,6), (7,8), (7,9), (7,10)\}.$$

$\square$*Exemple*

Soit $L$ un langage défini par une formule de $(IMP_2)^{Match}$. Soient $w = w_1 \dots w_n$ et $M$ le couple formé d'un mot de $L$ et d'un appariement $M$ sur les positions de ce mot, voir pour l'exemple figure 3.2. Construisons un arbre binaire, $T_{w,M}$, pour chaque mot $w \in L$.

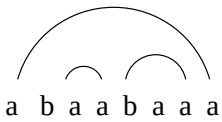



FIG. 3.2 – Un mot $w$ et un appariement $M$

1. On commence par construire un arbre $\widetilde{T}_{w,M}$ dont les nœuds sont les positions de $w$ et les arcs de $M$ (figure 3.3).

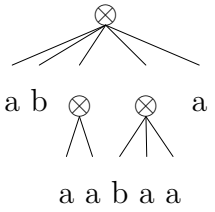



FIG. 3.3 – L'unique arbre $\widetilde{T}_{w,M}$ construit à partir de la figure précédente

   - Pour chaque $1 \leq i \leq n$, $i$ est une feuille étiquetée $w_i$ ;
   - un arc $(i, j) \in M$ est un nœud interne étiqueté $\otimes$, ses fils sont toutes les positions $k$ et arcs $(k, l) \in M$ $i < k < l < j$ dans l'ordre de leurs apparitions dans $(w, M)$.

2. À chaque fois qu'un nœud interne admet plus de deux fils, on les distribue en sous-arbres binaires, en utilisant des nœuds additionnels étiquetés $\odot$ comme suit : tous les nœuds $\odot$ sont des fils gauches des nœuds internes. Concrètement, on construit à partir d'un nœud étiqueté $\otimes$, un sous arbre en "peigne". Cette méthode engendre un unique arbre binaire $T_{w,M}$ dont le mot est $w$ et dont les nœuds internes sont étiquetés $\otimes$ ou $\odot$ (figure 3.4).

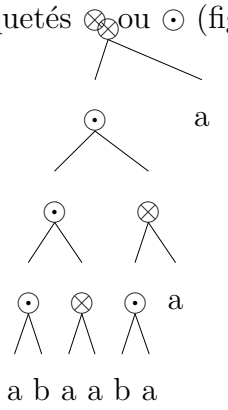



FIG. 3.4 – L'unique arbre binaire $T_{w,M}$ construit à partir de $\widetilde{T}_{w,M}$

Les arbres ainsi construits sont tels que :
– les feuilles sont étiquetées par des symboles de $\Sigma$ ;
– les nœuds internes sont étiquetés par $\otimes$ ou $\odot$.
Parmi tous les arbres satisfaisant ces deux conditions, les arbres $T_{w,M}$ construits à partir de $(S_w, M)$ doivent satisfaire la propriété suivante :
– Le chemin le plus à gauche et le chemin le plus à droite issus d'un nœud étiqueté $\otimes$ mènent à des feuilles, sans passer par aucun autre nœud étiqueté $\otimes$.
Cette propriété peut être exprimée dans la logique monadique du second ordre sur la signature :

$$(\{P_a, a \in \Sigma\}, P_\otimes, P_\odot, <, C).$$

Cette formule s'écrit :

$$\begin{gathered}\forall x[P_\otimes(x) \to \exists y \exists X \forall s \forall t(Pt(X,x,y) \wedge ((X(s) \wedge X(t) \wedge C(s,t)) \to Lc(s,t)) \wedge \\ \wedge((X(s) \wedge P_\otimes(s)) \to s = x))] \wedge \\ \wedge \forall x[P_\otimes(x) \to \exists y \exists X \forall s \forall t(Pt(X,x,y) \wedge ((X(s) \wedge X(t) \wedge C(s,t)) \to Rc(s,t)) \wedge \\ \wedge((X(s) \wedge P_\otimes(s)) \to s = x))].\end{gathered}$$

On ajoute la condition :

$$\forall i((P_\odot(i) \vee P_\otimes(i)) \to (\forall j(Rc(i,j) \to (Lf(j) \vee P_\otimes(j))))),$$

pour avoir un unique arbre de dérivation à partir de chaque appariement, comme expliqué en 2 plus haut.

**Remarque.** Cette formule nous assure de la construction d'un unique arbre pour chaque appariement, alors que, dans la preuve de [LST95], à partir d'un appariement on peut construire plusieurs arbres de dérivation de ce mot. □

Notons $\Phi_1$ la conjonction de ces deux propriétés du second ordre monadiques.

Par hypothèse, $(S_w, M) \models \phi$ pour une formule du premier ordre $\phi$ qui définit $M$ implicitement. Étant donné l'arbre de dérivation $T$, on peut retrouver facilement l'appariement $M_T$ qui l'a induit comme suit :

> $(i, j) \in M_T$ *si, et seulement si, l'ancêtre commun* $c$ *à* $i$ *et* $j$ *dans* $T$ *est étiqueté* $\otimes$ *et* $i$ *se trouve sur le chemin le plus à gauche et* $j$ *sur le chemin le plus à droite à partir de* $c$.

On veut maintenant que $T$ vérifie $(S_w, M_T) \models \phi$. Ceci est donné par la formule $\Phi_2$ qui est déduite de $\phi$ comme suit :

– on restreint tous les quantificateurs à borner uniquement les feuilles ; pour cela on remplace chaque sous-formule de la forme $\exists x\, \chi$ par $\exists x(Lf(x) \wedge \chi)$, et chaque occurrence de $\forall x\, \chi$ par $\forall x(Lf(x) \rightarrow \chi)$ ;
– on remplace $x < y$ par la formule du second ordre monadique :

$$\exists n\, l\, r(Lc(l, n) \wedge Rc(r, n) \wedge (l = x \vee An(l, x)) \wedge (r = y \vee An(r, y)))\,;$$

– on remplace les occurrences de $M(x, y)$ par une formule monadique du second ordre qui dit qu'il existe un nœud $n$ étiqueté $\otimes$, tel que le chemin le plus à gauche partant de $n$ mène à $x$ et le chemin le plus à droite mène à $y$.

Il existe donc un appariement $M$ sur le mot $w$ tel que $(S_w, M) \models \phi$ si, et seulement si, $w$ est le mot de $T$ pour un arbre $T \models (\Phi_1 \wedge \Phi_2)$. L'appariement qu'on utilise est implicitement défini par $\phi$, donc unique, et l'arbre construit à partir de cette structure de mot est uniquement déterminé à partir de l'appariement $M$. On déduit que :

*Étant donné un langage* $L$, *constitué des mots qui satisfont une certain énoncé de* $(IMP_2)^{Match}$, *on peut construire une classe d'arbres* $\tau$ *telle que :*

1. *chaque mot* $w$ *de* $L$ *correspond à un unique arbre* $T_w$ *de* $\tau$ *tel que :*

$$Yield(T_w) = w \wedge \forall T \in \tau(Yield(T) = w \Rightarrow T = T_w)$$

2. *chaque arbre* $T$ *de* $\tau$ *correspond à un unique mot* $w \in L$ *tel que :*

$$Yield(T) = w$$

3. *$\tau$ est un langage d'arbres régulier puisqu'il satisfait un énoncé de la logique monadique du second ordre.*

Puisque $\tau$ est un langage d'arbres régulier, il existe une grammaire régulière

$$G = (A, N, \{\odot, \otimes\} \cup \Sigma, R)$$

telle que $\mathcal{L}(G) = \tau$ et pour tout $T \in \tau$, il existe une unique dérivation, la plus à gauche, de $T$ dans $G$, et $R$ est constitué de règles de la forme :

$X \to \otimes(Y, Z)$

$X \to \odot(Y, Z)$

$X \to a$

pour des $X, Y, Z \in N$ et un $a \in \Sigma$, pour l'existance d'une telle grammaire voir [GS84] et Gécseg m'a communiqué l'existance d'une telle grammaire non-ambigüe [Géc01].

Dans la preuve du théorème de Mezei et Wright [MW67], la grammaire de mot associée à $G$ est la grammaire $G' = (A, N, \Sigma, R')$, telle que :

$X \to YZ \in R'$ pour $X \to \odot(Y, Z) \in R$ ou $X \to \otimes(Y, Z) \in R$

$X \to a \in R'$ pour $X \to a \in R$.

La grammaire de mots $G'$ est ambigüe si, et seulement si, on a une des deux conditions suivantes :

1. il existe $T$ et $T'$ dans $\tau$ tels que $Yield(T) = Yield(T')$, ce qui contredit la propriété 1 énoncée plus haut ;
2. ou on a deux dérivations diffèrentes pour un même arbre $T$ dans $G$, ce qui contredit la non-ambiguïté de $G$.

On peut donc conclure que les langages définis dans $(IMP_2)^{Match}$ sont algébriques non-ambigus.

## 3.5 Indécidabilité de $IMP_2$

Kolaitis [Kol90] a prouvé que l'ensemble $IMP$ est co-récursivement énumérable complet, donc indécidable. Nous allons montrer que déjà la logique utilisant la définition implicite d'un unique prédicat, qui est binaire, est indécidable dans le THÉORÈME 2. L'intérêt de ce résultat est son optimalité. On a vu dans la section 1.7 que la logique n'utilisant que des prédicats unaires définis implicitement est équivalente à $\exists MSO(<)$ qui caractérise la classe des langages réguliers, donc décidable. On prouve que la logique utilisant la définition implicite d'un seul prédicat, qui est binaire n'est pas décidable.

**THÉORÈME 2** *On ne peut pas décider si, oui ou non, un prédicat binaire est implicitement définissable.*

**Preuve :**

Le problème de savoir si un langage algébrique est non-ambigu est indécidable, voir par exemple [Har78]. D'après le théorème précédent, étant donné un appariement, on ne peut pas décider s'il est implicitement définissable, sachant que la propriété "être un appariement" est du premier ordre, voir page 49. $\Box Preuve$

On obtient aussi une nouvelle preuve du résultat de Gurevich [Gur84] selon lequel le théorème de définissabilité de Beth n'est plus vérifié si on se restreint aux structures de mots.

**Corollaire 3.1** *La propriété de Beth n'est pas vérifiée si on se restreint aux structures de mots finis.*

**Preuve :**

Sachant (voir [Har78, ABB97]) qu'il existe des langages algébriques intrinsèquement ambigus, on a des langages définissables dans $\exists Match\ FO$ qui ne sont pas définissables dans $(IMP_2)^{Match}$. $\Box Preuve$

# Chapitre 4

# Langages des réseaux de Petri et complexité

**Résumé**

On va prouver que les langages reconnus par des réseaux de Petri sont strictement inclus dans $NTime[n]$ (THÉORÈME 5). On prouvera aussi que les langages algèbriques sont dans la clôture du premier ordre de $\exists\mathbb{PP}$ (THÉORÈME 4). On en déduira que la logique $\exists\mathbb{PP}$ est strictement moins expressive que $FO(\exists\mathbb{PP})$ (THÉORÈME 3).

Dans la section 2 du chapitre des préliminaires, on a rappelé la définition d'une classe de langages basée sur les réseaux de Petri, notée $PNL$. Comparée aux autres classes de complexité de la hiérarchie de Chomsky, il a été prouvé que cette classe se positionne comme suit :

$$Reg \subsetneq PNL \subseteq CS = NSpace[n],$$

voir par exemple le livre de Peterson [Pet81] pour les preuves.

On va raffiner la seconde inclusion en prouvant que $PNL \subsetneq NTime[n]$.
La preuve de cette inclusion stricte utilise une comparaison des pouvoirs d'expressions de deux langages logiques : le premier, défini par Parigot et Pelz [PP85], que nous noterons $\exists\mathbb{PP}$ (voir section 2.7), caractérise $PNL$ ; le second, défini par Lautemann et *al* [LSS99], caractérise $NTime[n]$ (voir section 2.8).

La prochaine section sera consacrée à la preuve de deux résultats.
Le premier, se situe dans la lignée des résultats de [Mat98, AFS00]. On prouve que le fragment existentiel d'une logique monadique du second ordre sur les structures de mots n'est pas close par quantifications du premier ordre.

Le second essaye de comprendre le lien, étrange, qu'il y a entre les langages algébriques et les langages des réseaux de Petri, tous les deux basés sur les langages de Dyck, et ne sont pas clos par complement. On prouve que les langages algébriques sont inclus dans une clôture (logique) des langages des réseaux de Petri.

## 4.1 Clôture du premier ordre de $\exists\mathbb{PP}$, langages algébriques et $NTime[n]$

**Définition 4.1** *On appelle* la clôture du premier ordre de $\exists\mathbb{PP}$*, et on note :*

$$FO(\exists\mathbb{PP}),$$

*les formules de* $\mathbb{PP}$ *dans lesquelles on autorise les alternances entre les quantificateurs du premier ordre et les quantificateurs existentiels monadiques. Par exemple, des alternances du type :*

$$\exists X \forall x \exists Y \cdots$$

*entre les variables monadiques du second ordre* $X$ *et* $Y$*, et la variable du premier ordre* $x$*, sont autorisées.*

Ajtai et *al*, et Matz [AFS00, Mat98] ont commencé l'étude de l'alternance entre quantificateurs du premier ordre et quantificateurs existentiels du second ordre. Ils ont prouvé que la clôture du premier ordre de la logique existentielle monadique du second ordre a un pouvoir d'expression strictement supérieur à la logique existentielle monadique du second ordre sur des signatures ayant au moins une relation binaire. D'autre part, sur les structures de mots et les structures d'arbres, Büchi et Doner [Büc60, Don70] ont prouvé que $MSO = \exists MSO$, voir par exemple [Tho97], donc :

$$\exists MSO = FO(\exists MSO) = MSO.$$

Nous allons prouver dans cette section ce qu'il en est avec les prédicat du second ordre $=_g$ et $\leq_g$. On établira le résultat suivant :

**THÉORÈME 3** *Sur les structures de mots finis, on a :*

$$\exists\mathbb{PP} \subsetneq FO(\exists\mathbb{PP}).$$

Pour démontrer ce résultat, commençons par prouver que les langages algébriques sont définissables dans $FO(\exists\mathbb{PP})$.

**THÉORÈME 4** *Sur les structures de mots finis, on a :*

$$CFL \subseteq FO(\exists\mathbb{PP}).$$

**Preuve : (du THÉORÈME 4)**

D'après le résultat de [LST95], caractérisant la classe des langages algébriques comme la classe des modèles des énoncés de la logique $\exists Match\, FO$, il nous suffit de prouver que les formules de $\exists Match\ FO$ sont exprimables dans $FO(\exists\mathbb{PP})$.

Soit $L$ un langage algébrique. Il existe une formule :

$$\Phi_L \equiv \exists M \phi(M),$$

qui définit les mots de $L$.

La formule $\alpha_M$ :

$$\exists X_1 X_2 (X_1 =_g X_2 \wedge \forall x \neg (X_1(x) \wedge X_2(x))),$$

du langage $FO(\exists\mathbb{PP})$, nous assure de l'existence de deux sous-ensembles disjoints de positions qui peuvent être interprétés, respectivement, comme l'ensemble des parenthèses ouvrantes :

$$X_2 \equiv \{x | \exists y M(x,y)\},$$

et l'ensemble des parenthèses fermantes :

$$X_1 \equiv \{x | \exists y M(y,x)\},$$

d'un appariement $M$.

On exprime ensuite, dans le langage $FO(\exists\mathbb{PP})$, le fait que deux positions, $x$ et $y$, sont liées par l'appariement $M$, formé par les ensembles $X_1$ et $X_2$, grâce à la formule $\mu(x,y)$ :

$$\exists Y_1 Y_2 \forall z [(Y_1(z) \leftrightarrow (X_1(z) \wedge x < z < y))$$
$$\wedge (Y_2(z) \leftrightarrow (X_2(z) \wedge x < z < y)) \wedge (X_2(x) \wedge X_1(y) \wedge Y_1 =_g Y_2)].$$

Il s'agit de la traduction d'un algorithme simple pour trouver la parenthèse fermante correspondant à une parentèse ouvrante.

Pour avoir uniquement des quantificateurs du second ordre existentiels, les occurrences de la formule atomique $M(x,y)$ doivent être toutes positives. On exprime pour cela $\neg M(x,y)$ par une formule positive en $M$. On peut écrire :

$$\neg M(x,y) \equiv \neg X_2(x) \vee \neg X_1(y) \vee \exists z (z \neq y \wedge M(x,z)).$$

On commence alors par remplacer les occurences négatives de $M(x,y)$ dans la formule initiale par leur traduction ne faisant figurer que des occurences positives de

$M$ comme donné plus haut. Ensuite, on remplace $\exists M$ par $\alpha_M$, puis les occurences de $M(x, y)$ par $\mu(x, y)$. On peut vérifier sans peine que ce processus produit des formules de $FO(\exists\mathbb{PP})$. $\square Preuve$

On peut déduire de ce résultat, et de celui de Book et Greibach [BG70, LSS99], disant que les langages reconnus en temps non déterministe linéaire sont exactement les images par des projections des intersection finies de langages algébriques, que même $NTime[n]$ est dans cette clôture.

**Corollaire 4.1** *Sur les structures de mots finis, on a :*

$$NTime[n] \subseteq FO(\exists\mathbb{PP}).$$

**Preuve : (du Corollaire)**

On sait que

$$NTime[n] = \exists M_1 \exists M_2 \exists M_3 \exists \bar{R}(\phi_1 \wedge \phi_2 \wedge \phi_3),$$

où, pour $i = 1,\ 2,\ 3$, $\phi_i \in FO[M_i, \bar{R}]$, où les $M_i$ sont restreints à être des appariements et les $R_i$ sont unaires, voir théorème 2.12. Par le théorème précédent, et la clôture de $FO(\exists\mathbb{PP})$ par conjonction et par quantification monadique existentielle, on obtient le résultat demandé. $\square Preuve$

**Preuve : (du THÉORÈME 3)**

On a $CFL \not\subset PNL$, voir par exemple [Pet81], On a aussi, d'après les résultats de [LST95] et [PP85], $\exists Match\ FO \not\subset \exists\mathbb{PP}$. D'où, en utilisant le THÉORÈME 4 :

$$CFL = \exists Match\ FO \subseteq FO(\exists\mathbb{PP}).$$

On en conclut :

$$\exists\mathbb{PP} \subsetneq FO(\exists\mathbb{PP}).$$

$\square Preuve$

## 4.2 $PNL \subsetneq NTime[n]$

On va améliorer une borne supérieure sur les langages des réseaux de Petri en termes de complexité sur les machines de Turing. La borne supérieure connue est $PNL \subseteq NSpace[n] = CS$, voir [Pet81, Pap94]. On va prouver ici que cette borne peut être améliorée en $NTime[n]$.

**THÉORÈME 5** *Sur les structures de mots finis, on a :*

$$PNL \subsetneq NTime[n].$$

**Preuve :**

Soit $L$ un langage de réseau de Petri. Il existe une formule $\exists \overline{X} \Phi$ qui caractérise $L$, où $\Phi$ est de la forme donnée dans le résultat de Parigot et Pelz, à savoir que $\Phi$ est une conjonction de formules de la forme $X =_g Y$ et d'une formule ne contenant pas d'occurrences de $=_g$ ni de $\leq_g$, voir [PP85] pages 162-164.

On va commencer par prouver que $X =_g Y$ est définissable par un énoncé de la forme $\exists M \in Match\ \psi$. On remarque que :

$$X =_g Y \Leftrightarrow (X - Y =_g Y - X).$$

Il suffit alors de définir un appariement entre les éléments de $Y - X$ et ceux de $X - Y$, sans se soucier des éléments de l'intersection. On donne alors la formule :

$$\exists M \forall x(((X(x) \wedge \neg Y(x)) \leftrightarrow \exists y M(y,x)) \wedge ((Y(x) \wedge \neg X(x)) \leftrightarrow \exists y M(x,y))).$$

Cette formule est équivalente à la formule $X =_g Y$. Elle dit que les éléments de $Y - X$ forment l'ensemble des parenthèses ouvrantes, et les éléments de $X - Y$ l'ensemble des parenthèses fermantes, d'un mot bien parenthèsé.

On va vérifier ensuite que notre formule, dans laquelle on remplace les formules de la forme $X =_g Y$ par leurs définitions en utilisant des appariements, satisfait les conditions du téorème 2.12.

$\Phi$ ne contient que des occurrences positives de $=_g$. Plus précisemment, elle est de la forme :

$$\exists \bar{R}(\phi \wedge \bigwedge_{p \in P} S_p =_g E_p),$$

pour des sous-ensembles $S_p,\ E_p$ parmis les éléments de $\bar{R}$.

Puisque les occurences de $=_g$ sont toutes positives, les quantificateurs d'appariements sont toujours existentiels. On remarque de plus que la traduction de $X =_g Y$ est une formule qui n'a que $X$ et $Y$ comme variables libres. On peut donc mettre les $\exists M$ dans le préfixe de quantificateurs, ce qui donne une formule de la forme :

$$\exists \overline{T}\ \Phi'$$

où $\Phi' \in FO$, et où les $T_i$ sont ou bien des prédicats unaires ou bien des relations binaires qui sont des appariements. Puisqu'on n'a que des variables du second ordre

quantifiées existentiellement, on peut les permuter sans modifier la sémantique de la formule. On met alors les variables binaires devant les variables unaires pour obtenir la forme recherchée.

Il ne nous reste plus qu'a prouver que la formule $\Phi'$ est équivalente à une conjonction de formules de la forme appropriée. Chaque appariement $M_i$ figure dans la définition d'une seule formule atomique de la forme $X_i =_g Y_i$. On obtient donc une formule de forme :

$$\exists \bar{M} \exists \bar{R} (\bigwedge_{p \in P} \phi_p),$$

où les éléments de $\bar{M}$ sont indexés par les places, et chaque $M_p$ ne figure que dans une seule formule $\phi_p$. On en conclut l'inclusion :

$$PNL \subseteq NTime[n].$$

Puisqu'il existe des langages algébriques qui ne sont pas dans $PNL$, voir [Pet81], et que $CFL \subseteq NTime[n]$, voir [BG70, LSS99], on déduit que cette inclusion est stricte.

$\square Preuve$

# Chapitre 5

# Une approche par la complexité descriptive de la hiérarchie linéaire

**Résumé**

On va donner quatre nouvelles caractérisations logiques de la classe des langages de la hiérarchie linéaire. Ces caractérisations seront basées sur la logique monadique du second ordre augmentée de quantificateurs généralisés, à savoir $=_g$, $\leq_g$, le quantificateur de Rescher, le quantificateur de Härtig et le quantificateur de majorité.

**Notation :** Soit $X$ un ensemble, on notera $|X|$ la cardinalité de cet ensemble. □

Un des plus célèbres problèmes se rapportant aux langages de la hiérarchie linéaire concerne les problèmes de *comptage*. Le problème général est énoncé comme suit :

> *Soit $R$ une relation rudimentaire (unaire) sur les entiers. Est-ce que la relation (binaire) définie par :*
>
> $$C_R(x, y) \equiv (y = |\{i < x | R(i)\}|),$$
>
> *est rudimentaire ?*

L'intérêt porté à ce problème vient du fait que la réponse positive générale est équivalente à plusieurs autres problèmes ouverts de la théorie de la complexité, voir par exemple [PW86, DLM98, EM98, HP93]. Paris et Wilkie [PW86] ont prouvé le résultat suivant, voir aussi [DLM98] pour une preuve plus proche de notre travail.

**Résultat 5.1** *Soit $R$ une relation rudimentaire. Si pour tout $x \in \mathbb{N}$ le nombre*

*d'éléments de $R$ inférieurs à $x$ est borné par une fonction polylogarithmique en $x$ alors la relation $C_R$ est rudimentaire.*

*Formellement, pour tout $k \in \mathbb{N}$ fixé, la relation de comptage :*

$$C_R(x, y) \equiv (y = |\{i < x | R(i)\}| \wedge y \leq (log_2(x))^k),$$

*est rudimentaire.*

On va, dans ce chapitre, prouver que si la logique monadique du second ordre possède un certain pouvoir de comptage, assez restreint, on arrive à caractériser logiquement la classe des langages rudimentaires. Ce pouvoir de comptage est donné sous la forme de *quantificateurs généralisés* de comparaison de cardinalité. Dans la section 1, on donne une première caractérisation basée sur la logique introduite par Parigot et Pelz [PP85]. Dans la section 2, trois autres variantes de cette logique sont données. Ces variantes sont basées sur les quantificateurs généralisés de Härtig, de Rescher et de Majorité.

## 5.1 $\mathbb{PP}$ vs. $RUD$

On va prouver, dans cette section, que la logique $\mathbb{PP}$ caractérise logiquement la classe $RUD$. On utilisera, pour prouver ce résultat, le théorème de Lynch-Immerman [Imm87, Lyn82] énoncé dans la Section 1.10, à savoir $MSO(+) = RUD$. Il nous suffit donc de comparer le pouvoir d'expression de $MSO(+)$ à celui de $\mathbb{PP}$.

**THÉORÈME 6** *Sur les structures de mots binaires finis, on a :*

$$\mathbb{PP} = MSO(+) = RUD = LinH.$$

La preuve de ce théorème va être divisée en deux parties. La première, la plus facile, sera de prouver que $MSO(+) \subseteq \mathbb{PP}$. La seconde sera la preuve de l'autre direction ; elle sera établie par une succession de lemmes.

### 5.1.1 Preuve de$(MSO(+) \subseteq \mathbb{PP})$

Pour prouver l'inclusion $(MSO(+) \subseteq \mathbb{PP})$, il nous suffit de donner une définition, dans la logique $\mathbb{PP}$, de la relation ternaire d'addition, notée $Add(x, y, z) \equiv (x + y = z)$. Soit la formule :

$$\exists XY(\forall t(X(t) \leftrightarrow t < y) \wedge \forall t(Y(t) \leftrightarrow x \leq t < z) \wedge Y =_g X).$$

Cette formule $Add(x, y, z)$ définit bien le prédicat $(x+y = z)$. En effet, cette formule dit simplement qu'il existe autant d'éléments avant la position $y$ que d'éléments entre les positions $x$ et $z$. Alors, étant donnée une formule de $MSO(+)$, il nous suffit d'y remplacer les occurrences de la formule atomique $(x + y = z)$ par sa définition $Add(x, y, z)$ pour obtenir une formule de $\mathbb{PP}$ qui lui est logiquement équivalente.

### 5.1.2 Preuve de $(MSO(+) \supseteq \mathbb{PP})$

La preuve de cette inclusion demande quelques lemmes.

**Lemme 5.1 ([Lyn82])** *La multiplication $\times$ est définissable dans le fragment existentiel de $MSO(+)$. On obtient donc :*

$$MSO(+) \equiv MSO(<, +, \times).$$

**Définition 5.1** *Soit le predicat $Bit(i, j)$ ssi le $i^{ème}$ chiffre de la numération binaire de $j$ est un 1.*

**Exemple.** La numération binaire de 9 est '1001'. On a donc $Bit(0, 9)$, $Bit(3, 9)$ et $\neg Bit(1, 9)$, $\neg Bit(2, 9)$. $\square Exemple$

**Lemme 5.2 ([Imm99])**

$$FO(<, +, \times) \equiv FO(<, Bit).$$

**Corollaire 5.1**

$$MSO(+) \equiv MSO(+, \times, <) \equiv MSO(<, Bit).$$

**Définition 5.2** *Soit $a$ un entier non nul. On associe à $a$, la structure de mot $S_a = ([n], <, A, Bit)$, représentant l'entier $a$, i.e. $S_a$ est la structure de mot associée à la numération binaire de $a$ ; avec $n$ la longueur de la décomposition binaire de $a$, $A$ le prédicat unaire contenant les positions 1 de la décomposition binaire de $a$, et $Bit$ la relation définie ci-dessus.*

**Définition 5.3** *Soit $a$ un entier non nul. On associe à $a$ la $\{<, c, Bit\}$-structure logique :*

$$\Omega_a = ([2^n], <, a, Bit),$$

*avec $n$ la longueur de la décomposition binaire de $a$. On appelle $\Omega_a$* la structure de représentation de $S_a$.

On remarque que la structure $\Omega_a$ n'est pas une structure de mot. On identifie les sous-ensembles de $S_a$ aux individus (éléments) de $\Omega_a$ de telle façon à ce que $X \subseteq [n]$ soit représenté par $x \in \Omega_a$ si, et seulement si, la fonction caractéristique de $X$ correspond à la numération binaire de $x$, c'est-à-dire :

$$x = \sum_{i \in X} (2^i).$$

Commençons par donner un exemple de traduction entre ces structures, avant d'énoncer un lemme qui nous assure toute la traduction.

**Exemple.** On définit l'addition, comme dans [Imm99], bit à bit en utilisant l'algorithme connu sous le nom de"carry lookahead algorithm". On définit le prédicat qui représente la retenue, $Carry$ :

$$Carry(i) \equiv \exists j(i > j \wedge X(j) \wedge Y(j) \wedge \forall k(j < k < i \rightarrow (X(k) \vee Y(k))).$$

On note $\oplus$ le connecteur *ou exclusif*.

On peut déduire l'addition sur les sous-ensembles de $S_a$ de la façon suivante :
$Z = X +_S Y$ si, et seulement si :

$$Z(i) \leftrightarrow (X(i) \oplus Y(i) \oplus Carry(i)).$$

On peut facilement vérifier qu'on a bien l'addition sur $\Omega_a$. En associant $x, y, z$ respectivement à $X, Y, Z$, on obtient :

$$\Omega_a \models (x + y = z) \Longleftrightarrow S_a \models (X +_S Y = Z).$$

$\square Exemple$

Énonçons maintenant deux lemmes de traduction entre les deux structures $S_a$ et $\Omega_a$.

**Lemme 5.3** ($MSO \rightarrow FO$) *On peut associer à toute formule monadique du second ordre $\Phi$ sur la signature des mots binaires $\{<, Bit, P\}$, une formule du premier ordre $\phi$ sur la signature $\{<, Bit, c\}$ telle que pour toute structure de mot $S_a$ et sa structure de représentation $\Omega_a$ associée on a :*

$$\Omega_a \models \phi \Longleftrightarrow S_a \models \Phi.$$

**Preuve :**

On commence par traduire les formules atomiques de $MSO(Bit, <, P)$. Supposons que $\alpha$ est une formule atomique de $MSO(Bit, <, P)$, alors $\alpha$ est dans l'une des formes suivantes :

1. $i < j$ pour deux variables individuelles $i$ et $j$ ;
2. $Bit(i, j)$ pour deux variables individuelles $i$ et $j$ ;
3. $X(i)$ pour une variable individuelle $i$ et une variable monadique du second ordre $X$ ;
4. $P(i)$ pour une variable individuelle $i$.

Remarquons que la longueur de la numération binaire est définissable dans $FO(Bit, <)$ :

$$Lg(j) = i \Leftrightarrow Bit(i, j) \wedge \forall k(Bit(k, j) \rightarrow k \leq j)$$

On traduit chacune des formules atomiques énoncée par :

– $i < j$ sur $S_a$ correspond à $i < j$ sur $\Omega_a$ ;
– $Bit(i, j)$ sur $S_a$ correspond à $Bit(i, j)$ sur $\Omega_a$ ;
– $X(i)$ sur $S_a$ correspond à $Bit(i, x)$ sur $\Omega_a$, où $x$ représente $X$ ;
– $P(i)$ sur $S_a$ correspond à $Bit(i, c)$ sur $\Omega_a$.

Pour retrouver la formule $\phi$ correspondant à $\Phi$, on procède comme suit :

1. on commence par remplacer les formules atomiques par leurs traductions données plus haut ;
2. on remplace les occurrences des variables du second ordre quantifiées par leurs variables correspondantes quantifiées de la même façon, *i.e.* $\forall X, \exists X$ sont remplacés par $\forall x, \exists x$ ;
3. on remplace les occurrences quantifiées des variables individuelles par ces même variables quantifiées avec une borne qui est la taille de la numération binaire de $a$, *i.e.* $\forall x, \exists x$ par $\forall x \leq Lg(c), \exists x \leq Lg(c)$.

On obtient ainsi la formule recherchée. L'idée générale est d'identifier les sous-ensembles de $[n]$ aux individus de $[2^n]$ et d'identifier les individus de $[n]$ aux individus du segment initial de taille $n$ de $[2^n]$, sachant qu'on peut définir $n$ dans la structure dont l'univers est $[2^n]$. $\square Preuve$

**Lemme 5.4** ($FO \rightarrow MSO$) *On peut associer, à toute formule du premier ordre $\phi$ sur la signature $\{<, Bit, c\}$, une formule monadique du second ordre $\Phi$ sur la signature des mots binaires $\{<, Bit, P\}$ telle que, pour toute structure de mot $S_a$ et sa structure de représentation $\Omega_a$ associée, on a :*

$$\Omega_a \models \phi \Longleftrightarrow S_a \models \Phi.$$

**Preuve :**

La construction de $\Phi$ se fait en remplaçant les variables du premier ordre de $\phi$ par les variables monadique du second ordre correspondantes, et en traduisant ensuite $Bit$ et $<$ par les définitions correspondantes $BITSET$ et $<_S$.

Les formules atomiques de $FO(Bit, <, c)$ sont de l'une des formes :
– $x < y$ ;
– $Bit(x, y)$ ;
– $c < x$ ;
– $x < c$ ;
– $Bit(c, x)$ ;
– $Bit(x, c)$.
Leurs formules correspondantes de $MSO(Bit, <, P)$ sont :
– $X <_S Y \equiv \exists x(Y(x) \wedge \neg X(x) \wedge \forall y(y > x \rightarrow (X(y) \leftrightarrow Y(y)))$ ;
– $BITSET(X, Y) \equiv \exists y(X(y) \wedge \forall t(Y(t) \leftrightarrow Bit(t, y)))$ ;
– $P <_S X$ ;
– $X <_S P$ ;
– $BITSET(P, X)$ ;
– $BITSET(X, P)$.
En procédant par induction sur la construction des formules, on peut prouver que la traduction s'adapte aux opérations booléennes.

Finalement, si on quantifie les variables monadiques du second ordre de $\Phi$ comme leurs correspondantes de $\phi$, on obtient la formule recherchée. $\square Preuve$

**Exemple.** Soit $\exists x \forall y(\neg x = y \rightarrow x < y)$ la formule qui définit 0 sur $\Omega_a$. Elle est traduite par $\exists X \forall Y(\neg X = Y \rightarrow X <_S Y)$, qui définit bien $\emptyset$ sur $S_a$. $\square Exemple$

**Remarque :** Les formules construites par la traduction dans les preuves des lemmes 5.3 et 5.4, vérifient les propriétés suivantes :

1. pour le lemme 5.3, si on part d'une formule de la forme :

$$Q_1 X_1 \cdots Q_k X_k \psi,$$

où les $Q_i$ sont des quantificateurs et $\psi$ est une formule du premier ordre, on construit une formule de la forme :

$$Q_1 x_1 \cdots Q_k x_k \psi',$$

où $\psi'$ est une formule du premier ordre dans laquelle tous les quantificateurs sont bornés par $|c|$ ;

2. pour le lemme 5.4, si on part d'une formule de la forme :

$$Q_1 x_1 \cdots Q_k x_k \theta,$$

où $\theta$ est une formule ouverte du premier ordre, on construit une formule de la forme :

$$Q_1 X_1 \cdots Q_k X_k \psi,$$

où $\psi$ est une formule du premier ordre.

3. On pouvait, dans le lemme 5.4 traiter de façons différentes les variables dont les quantificateurs sont bornés par $|c|$ et les autres variables non bornées. Dans un recent travail [Hac01], j'ai prouvé une équivalence entre les résultats de Lynch-Immerman [Lyn82, Imm87] et les résultats de Kent et Hodgson [KH82, KH83], en utilisant cette dernière traduction.

Le dernier lemme dont on a besoin dans cette preuve est appelé par Immerman (The Bit Sum Lemma).

**Définition 5.4** *Soit $BSUM(x, y)$, le prédicat binaire qui est vrai ssi $x$ est égal au nombre de 1 dans la numération binaire de $y$.*

**Lemme 5.5 ([BIS90, Imm99])** *$BSUM$ est définissable dans $FO(<, Bit)$.*

On peut maintenant conclure la preuve du théorème.

**Preuve :** $(MSO(+) \supseteq \mathbb{PP})$.

Pour cela il nous suffit de donner une formule qui définit $=_g$ et $\leq_g$ dans $MSO(+)$. D'après le (Bit Sum Lemma) et les lemmes de traduction (lemme 5.3 $MSO \to FO$) et (lemme 5.4 $FO \to MSO$), il existe une formule $BSUMSET(X, Y) \in MSO(+)$ telle que :

$$\Omega_a \models BSUM(x, y) \Longleftrightarrow S_a \models BSUMSET(X, Y).$$

Notons $X_{<t}$ le sous-ensemble de $X$ composé des éléments qui sont inférieurs à $t$. On peut donc définir le prédicat $X \leq_g Y$ par la formule de $MSO(<, Bit)$ :

$$\forall t \exists Z W (BSUMSET(X_{<t}, Z) \wedge BSUMSET(Y_{<t}, W) \wedge (W <_S Z \vee \forall i (W(i) \leftrightarrow Z(i))).$$

notée $\psi_{\leq_g}(X, Y)$, et le prédicat $X =_g Y$ par la formule :

$$\psi_{\leq_g}(X, Y) \wedge \exists U V (BSUMSET(X, U) \wedge BSUMSET(Y, V) \wedge \forall j (U(j) \leftrightarrow V(j)).$$

notée $\psi_{=_g}(X, Y)$.

Remarquons que $\psi_{\leq_g}(X, Y)$ dit simplement que, pour toute position $t$, il existe plus d'éléments inférieurs à $t$ dans $Y$ que d'éléments inférieurs à $t$ dans $X$. La formule $\psi_{=_g}(X, Y)$ dit qu'on a $\psi_{\leq_g}(X, Y)$ et qu'à la fin on a autant d'éléments dans $X$ que d'éléments dans $Y$.

Étant donnée une formule de $\mathbb{PP}$, il nous suffit d'y remplacer les occurrences de la formule atomique $X \leq_g Y$ (respectivement $X =_g Y$) par sa définition $\phi_{\leq_g}(X, Y)$ (respectivement $\phi_{=_g}(X, Y)$) pour obtenir une formule de $MSO(+, \times, Bit, <)$ qui lui est logiquement équivalente. $\square Preuve$

## 5.2 Des variantes de $\mathbb{PP}$

On va, dans cette section, donner trois logiques que l'on prouvera équivalentes à $\mathbb{PP}$, donc caractérisant logiquement $RUD$. Cette section sera divisée en trois parties, chacune correspondant à une variante de $\mathbb{PP}$.

**Notation.** Notons $X_{<t}$ le sous-ensemble de $X$ composé des éléments qui sont inférieurs à $t$. □

### 5.2.1 Le quantificateur de Rescher et $RUD$

Le *quantificateur de Rescher*, introduit en 1962 [Res62] pour étudier les logiques multi-valuées, est un exemple de quantificateur généralisé, ou quantificateur de Lindström, voir par exemple [Vää97, EF95] pour une définition générale.

**Définition 5.5** *Le* quantificateur de Rescher*, noté $Q_r$, est défini par :*

$$Q_r(X,Y) \equiv |X| < |Y|,$$

*pour deux variables unaires $X$ et $Y$.*

*La logique $MSO(Q_r)$ est la logique monadique du second ordre dont les formules atomiques contiennent, outre les formules atomiques de $MSO(<)$, les formules atomiques de la forme $Q_r(X,Y)$ pour toutes variables monadiques $X, Y$.*

**Exemple.** Soit la signature $\tau = (<, P)$ des structures de mots binaires. La formule :

$$\exists XY \forall t[(X(t) \leftrightarrow P(t)) \wedge (Y(t) \leftrightarrow \neg X(t)) \wedge Q_r(Y,X)],$$

dit qu'il existe plus de 1 que de 0 dans le mot. □*Exemple*

On arrive maintenant à notre théorème.

**THÉORÈME 7** *Sur les structures de mots binaires finis, on a :*

$$MSO(Q_r) \equiv MSO(+) \equiv \mathbb{PP} \equiv RUD \equiv LinH.$$

**Preuve :**

La formule[1] :

$$\forall t \neg (Q_r(Y_{<t}, X_{<t})).$$

définit $X \leq_g Y$. La formule :

$$\forall t (\neg Q_r(Y_{<t}, X_{<t}) \wedge \neg (Q_r(X, Y) \vee Q_r(Y, X)),$$

définit $X =_g Y$ dans $MSO(Q_r)$. D'où la première inclusion :

$$\mathbb{PP} \subseteq MSO(Q_r).$$

Pour l'autre direction, on utilise le fait que $MSO(+) \equiv \mathbb{PP}$ peut compter le nombre d'éléments dans un ensemble quelconque en utilisant le prédicat $BSUMSET$. La formule :

$$\exists ZW (BSUMSET(X, Z) \wedge BSUMSET(Y, W) \wedge Z <_S W),$$

est une définition de $Q_r(X, Y)$ dans $\mathbb{PP} \equiv MSO(+)$. $\square Preuve$

[1]La définition de $MSO(Q_r)$ qu'on a donnée restreint l'utilisation de ce quantificateur à des variables. On peut, sans augmenter le pouvoir d'expression de cette logique appliquer $Q_r$ à des formules. Ceci est faisable en rajoutant des variables monadiques qu'on définit équivalentes aux formules données en arguments à $Q_r$.

### 5.2.2 Le quantificateur de Härtig et $RUD$

**Définition 5.6** *Le* quantificateur de Härtig *[Här65], noté $Q_h$, est défini par :*

$$Q_h(X,Y) \equiv |X| = |Y|,$$

*pour deux variables unaires $X$ et $Y$.*

*La logique $MSO(Q_h)$ est la logique monadique du second ordre dont les formules atomiques contiennent, outre les formules atomiques de $MSO$, les formules atomiques $Q_h(X,Y)$ pour toutes variables monadiques $X, Y$.*

**Exemple.** Sur les structures de mots binaires, la formule :

$$\exists XY \forall t[(X(t) \leftrightarrow P(t)) \wedge (Y(t) \leftrightarrow \neg X(t)) \wedge Q_h(X,Y)],$$

définit l'ensemble des entiers dont la numération binaire contient autant de uns que de zéros. $\square Exemple$

**THÉORÈME 8** *Sur les structures de mots binaires finis, on a :*

$$MSO(Q_h) \equiv MSO(Q_r) \equiv \mathbb{PP} \equiv MSO(+) \equiv RUD \equiv LinH.$$

**Preuve :**

D'après le théorème précédent, il suffit de donner une définition de $Q_r$ dans $MSO(Q_h)$ et une définition de $Q_h$ dans $MSO(Q_r)$.

Le quantificateur de Härtig, $Q_h(X,Y)$, est défini dans $MSO(Q_r)$ par :

$$(\neg Q_r(X,Y) \wedge \neg Q_r(Y,X)).$$

C'est exactement comme la définition naturelle de $=$ à partir de $<$.

Le quantificateur de Rescher, $Q_r(X,Y)$, est défini dans $MSO(Q_h)$ par :

$$\exists Z \forall t((X(t) \rightarrow Z(t)) \wedge \exists u(Z(u) \wedge \neg X(u)) \wedge Q_h(Z,Y)).$$

Cette formule dit simplement qu'il existe un sous-ensemble $Z$ qui contient strictement $X$ et qui a la même cardinalité que $Y$. $\square Preuve$

### 5.2.3 Le quantificateur de majorité et $RUD$

On va prouver, dans cette section, qu'en augmentant la logique monadique du second ordre avec un prédicat de majorité, on atteint la classe des rudimentaires. Pour ceci, on compare le pouvoir d'expression de la logique $MSO(Maj)$ à celui de $MSO(Q_r, Q_h)$.

On va commencer par donner une définition formelle du prédicat de *majorité*. Ce prédicat est un exemple typique des mécanismes de comptage. Il a été utilisé par Ajtai [Ajt83] pour séparer $AC^0$ (la classe des langages reconnus par des circuits booléens de taille polynomiale et de profondeur constante, voir par exemple [Pap94]) de $NC^1$ (la classe des langages reconnus par des circuits booléens de taille polynomiale, de profondeur polylogarithmique et d'entrée constante). L'inclusion stricte de $AC^0$ dans $NC^1$ a été prouvée par Furst, Saxe et Sipser [FSS84] en utilisant le prédicat de parité. Un mot binaire satisfait le prédicat de parité si, et seulement si, le nombre de 1 dans ce mot est pair.

**Définition 5.7** *Soit $X$ une variable unaire. On écrit $Maj(X)$ pour dire que le nombre d'éléments qui sont dans $X$ est strictement supérieur à la moitié du nombre total de positions dans le mot. (On a choisi la majorité stricte pour un but simplement technique, car la majorité large n'est que le dual de la majorité stricte.)*

Il pourrait sembler que $Maj$ soit strictement plus faible que les quantificateurs $Q_h, Q_r$ à cause de son arité : il ne prend qu'une seule variable en argument alors que $Q_h$ et $Q_r$ en prennent deux. On va ici prouver que cele n'est pas vrai, et que $Maj$ peut à lui tout seul définir toutes les relations définies auparavant en utilisant les quantificateurs $Q_h$ et $Q_r$, et donc toutes les relations définies à l'aide de l'addition, en présence de la logique monadique du second ordre.

**THÉORÈME 9** *Sur les structures de mots binaires finis, on a :*

$$MSO(Maj) \equiv MSO(Q_h) \equiv MSO(Q_r) \equiv \mathbb{PP} \equiv MSO(+) \equiv RUD \equiv LinH.$$

**Preuve :**

Pour prouver que $Maj(X)$ est définissable dans $MSO(Q_r)$, on considère la formule :

$$\exists Y(\forall t(Y(t) \leftrightarrow \neg X(t)) \wedge Q_r(Y, X)).$$

Cette formule dit qu'il y a moins de positions satisfaisant $\neg X$ que de positions satisfaisant $X$, d'où on a plus de la moitié des positions sont dans $X$.

Pour la seconde direction, supposons qu'on ait $Q_r(X, Y)$. Notons $n$ le nombre d'éléments dans la structure, qui correspond à $max + 1$ avec ($max$ définit par $\forall x(x \leq max)$). Trois cas (exclusifs) peuvent se présenter :

1. Supposons que $|X(x)| \leq \frac{n}{2} < |Y(x)|$. On peut facilement exprimer ceci dans $MSO(Maj)$ par :
$$Maj(Y) \wedge \neg Maj(X).$$

2. Supposons ensuite que $|X(x)| < |Y(x)| \leq \frac{n}{2}$. On doit alors rajouter autant d'éléments à $X$ qu'à $Y$, pour être dans le cas précédent. Ces éléments doivent être dans le complément de $X \cup Y$ pour ne pas augmenter la cardinalité d'un des ensembles sans augmenter la cardinalité de l'autre ; de tels éléments existent car $|X(x)| < |Y(x)| \leq \frac{n}{2}$ et donc $|X(x)| + |Y(x)| < n$. La formule de $MSO(Maj)$ :

   $\exists Z[(Z(x) \rightarrow (\neg X(x) \wedge \neg Y(x))) \wedge (\neg Maj(X(x) \vee Z(x)) \wedge Maj(Y(x) \vee Z(x))]$

   dit qu'il existe un entier positif $i$ tel que

$$i + |X(x)| \leq \frac{n}{2} < i + |Y(x)|,$$

   ce qui est équivalent à $|X(x)| < |Y(x)|$ qui est la définition de $Q_r(X, Y)$.

3. Supposons enfin que $\frac{n}{2} < |X(x)| < |Y(x)|$. On utilise l'équivalence entre cette inégalité et le fait :
$$\frac{n}{2} > |\neg X(x)| > |\neg Y(x)|,$$
   En utilisant deux nouvelles variables $Z \leftrightarrow \neg X$ et $T \leftrightarrow \neg Y$, on a $|T(x)| < |Z(x)| < \frac{n}{2}$ ce qui s'exprime dans $MSO(Maj)$ par le cas précédent.

On peut vérifier sans peine qu'on a $Q_r(X, Y)$ si, et seulement si, un et un seul des trois cas énumérés est satisfait.

On conclut donc que la disjonction de ces trois formules définit le quantificateur de Rescher dans $MSO(Maj)$. Ceci termine la preuve. $\Box Preuve$

Le résultat d'Ajtai [Ajt83] était que $Maj$ n'est pas définissable dans $FO(<)$. On peut maintenant l'étendre à : $Maj$ n'est pas définissable dans la logique monadique du second ordre (sans prédicats arithmétiques autres que l'ordre). Ce résultat est déja connu comme application directe du lemme de pompage (pumping lemma) pour les langages réguliers, sachant que $MSO(<) \equiv Reg$.

**Corollaire 5.2** *Le quantificateur de majorité n'est pas définissable dans la logique monadique du second ordre avec $<$ comme unique prédicat.*

**Preuve :**

On ne fait que rappeler les résultats de théorie des langages :

$$MSO(<) = Reg \subsetneq CFL \subsetneq NTime[n] \subseteq LinH = MSO(Maj, <).$$

Voir [Har78] pour la première inclusion, [BG70] pour la seconde et [Wra78] pour la troisième. Pour ce qui est des deux égalités, la première est le théorème de Büchi [Büc60] et la seconde n'est autre que le théorème précédent. $\Box Preuve$

Dans leur article [BIS90], Barrington, Immerman et Straubing ont défini le quantificateur de la majorité des paires, noté $Maj^2$ :
$Maj^2(\phi(x, y))$ si, et seulement si, pour chaque structure de cardinalité $n$, le nombre de couples $(x, y)$ satisfaisant $\phi$ est supérieur à $\frac{n^2}{2}$.

Ils ont prouvé que, dans $FO$, en utilisant $Maj$ et $Bit$, on peut définir $Maj^2$, et que, en l'absence de $Bit$, $Maj$ et $Maj^2$ définissent $Bit$. Récemment, il a été prouvé [LMSV98] que $Maj^2$ n'est pas définissable dans $FO$ en l'absence de $Bit$, en utilisant $Maj$.

**THÉORÈME 10** *Même en l'absence de Bit, on peut définir $Maj^2$ à partir de $Maj$ dans $MSO(<)$.*

Ce résultat est conséquence du théorème $MSO(Maj, <) \equiv MSO(Bit, <)$ et du fait que que $Maj^2$ est définissable à partir de $Maj$ et $Bit$ dans la logique du premier ordre, par conséquent, dans la logique monadique du second ordre.

## 5.3 Une approche descriptive de problèmes autour de $LinH$

On va, dans cette section, donner quelques équivalences entre des problèmes de hiérarchie en complexité et des problèmes de complexité descriptive. On conclura donc que ces problèmes sont du même ordre de "complexité".

**Définition 5.8** *Soit $\phi(x_1, \cdots, x_k, y_1, \cdots, y_k)$ une propriété $2k$-aire.*
– *On note $(TC_{x_1,\cdots,x_k,y_1,\cdots,y_k}\phi)$ la clôture transitive et réflexive de $\phi$.*
– *On note $(DTC_{x_1,\cdots,x_k,y_1,\cdots,y_k}\phi)$ la clôture transitive et réflexive de*

$$\phi(\bar{x}, \bar{y}) \wedge \forall \bar{z}(\neg\phi(\bar{x}, \bar{z}) \vee \bar{z} = \bar{y}).$$

**Théorème 5.1 ([Imm99], Theorem 10.27)** *Pour $k = 1, 2, \cdots$*

1. *$DSpace[n^k] = SO(arity\ k)(DTC)$ ;*
2. *$NSpace[n^k] = SO(arity\ k)(TC)$.*

**THÉORÈME 11** *Les trois propriétés suivantes sont équivalentes :*

1. *la logique $FO(Bit, <)$ est close par clôture transitive déterministe ;*
2. *la logique $MSO(Bit, <)$ est close par clôture transitive déterministe ;*
3. *$LinH = DSpace[n]$.*

Remarquons que $FO(Bit, <)$ n'est pas un langage logique de structures de mots. Les problèmes connus qui séparent $FO(Bit, <) + DTC$ de $FO(Bit, <)$ utilisent le prédicat unaire qui collecte les positions 1, comme la parité ou la majorité. À ma connaissance, aucun problème n'utilisant que les relations numériques $+$, $\times$, $Bit$ et $<$ séparant ces deux logiques n'a été annoncé. De plus, les résultats positifs sur le pouvoir de de comptage de $AC^0$ donnés dans [DGS86, AB84] sont hérités par la hiérerchie linéaire mais pas les résultats négatifs.

**Preuve :** (Du théorème 11)

La première équivalence $1 \Leftrightarrow 2$ est due aux deux lemmes (lemme 5.3 $FO \rightarrow MSO$) et (lemme 5.4 $MSO \rightarrow FO$), et une comparaison très facile des étapes pour la clôture transitive. La seconde équivalence est due à la caractérisation logique des $DSpace[n]$ donnée par Immerman, voir [Imm99] Theorem 10.27, qui dit que $DSpace[n] = MSO(DTC)(<)$. Sachant qu'on peut construire l'addition, donc tous les prédicats définissables dans $MSO(Bit, <)$, par $DTC$ et $<$, on a le résultat $2 \Leftrightarrow 3$. $\square Preuve$

**THÉORÈME 12** *Les trois propriétés suivantes sont équivalentes :*

1. *la logique $FO(Bit, <)$ est close par clôture transitive ;*
2. *la logique $MSO(Bit, <)$ est close par clôture transitive ;*
3. *$LinH = NSpace[n]$.*

La preuve est la même que le théorème précédent, en remplaçant $DTC$ par $TC$ et $DSpace[n]$ par $NSpace[n]$.

**Définition 5.9** *Soit $R(x)$ un prédicat. Le prédicat de comptage associé à $R$, noté $C_R(x, y)$, est l'ensemble des couples $(x, y)$ tels que :*

$$y = |\{i | i < x \text{ et } R(i)\}|.$$

*On notera $\#LinH$ la clôture de $LinH$ par le comptage.*

**THÉORÈME 13** *Les trois propriétés suivantes sont équivalentes :*

1. *la logique $FO(Bit, <)$ est close par comptage (ou $Maj$) ;*
2. *la logique $MSO(Bit, <)$ est close par comptage ;*
3. *$LinH = \#LinH$.*

**THÉORÈME 14** *Les trois propriétés suivantes sont équivalentes :*

1. *la logique $FO(Bit, <)$+$Comptage$ est close par clôture transitive déterministe ;*
2. *la logique $MSO(Bit, <)$+$Comptage$ est close par clôture transitive déterministe ;*
3. *$\#LinH = DSpace[n]$.*

Les preuves des deux théorèmes précédents sont similaires à celles des théorèmes 11 et 12, sachant que $\#LinH$ est caractérisée par $MSO(Bit, <) + Comptage$ et que le comptage est une forme de clôture transitive (récursion).

**THÉORÈME 15** *Si $TC^0 = L$ alors $\#LinH = DSpace[n]$.*

Le problème de savoir si $TC^0 = L$ est très célèbre en théorie de la complexité, voir [Ruh99] pour une discussion. On peut donc dire que ces problèmes sont de même ordre de difficulté.

# Chapitre 6

# Un jeu combinatoire pour $MSO(Q_h, Q_r)$

**Résumé**

On va donner un jeu combinatoire, basé sur les caractérisations logiques de $LinH$ données dans le chapitre précédent, qui est une méthode potentielle pour prouver l'appartenance ou non d'un problème à $LinH$. On donne, comme application de ce jeu, une extension du résultat de Schwentick sur la non définissabilité de la connexité des graphes finis dans $\exists MSO(<)$ en prouvant que, même en utilisant les quantificateurs de Härtig et de Rescher, cette propriété reste toujours non définissable. Enfin, on comparera le pouvoir d'expression de deux parties existentielles de $MSO$ avec les prédicats de Rescher et Härtig sur les structures de mots.

Après les résultats de Büchi [Büc60] et de Fagin [Fag74], la logique du second ordre est devenue un domaine de recherche très étudié par la communauté de la complexité descriptive. Ce qu'on veut savoir est :

1. si un langage est définissable dans une logique donnée. Il nous suffit pour cela d'exhiber une formule dans la logique en question qui définit ce langage pour confirmer la définissabilité ;
2. si un langage n'est pas définissable dans une logique donnée. Pour cela on a besoin d'outils pour prouver la non-définissabilité.

Des outils de non-définissabilité ont été donnés : lois 0-1, jeux, arguments de localité, voir [EF95, Imm99, Fag97]. Malheureusement, dans le cas des logiques du second ordre, aucune stratégie universelle n'est connue, voir
`http ://www-mgi.informatik.rwth-aachen.de/FMT/problems.ps`

## 6.1 Le jeu pour $MSO(Q_h, Q_r)$

**Définition 6.1** *Soient deux joueurs $I$ et $II$ comme dans la section sur les jeux de $FO$ et de $MSO$. Le joueur $I$ essaie de prouver que les structures diffèrent par une formule de la logique $MSO(Q_h, Q_r)$, et le joueur $II$ essaie de prouver le contraire. Soient $\mathcal{A}$ et $\mathcal{B}$ deux structures. Le jeu $HR_{m,n}(\mathcal{A}, \mathcal{B})$[1] se joue exactement comme le jeu $MSO_{m,n}(\mathcal{A}, \mathcal{B})$. Le tableau suivant représentera les choix des joueurs sur chaque structure, les majuscules représentant les ensembles et les minuscules représentent les individus :*

| $\mathcal{A}$ | $\mathcal{B}$ |
|---|---|
| $C_1$ | $C'_1$ |
| $C_2$ | $C'_2$ |
| $\vdots$ | $\vdots$ |
| $C_m$ | $C'_m$ |
| $c_1$ | $c'_1$ |
| $c_2$ | $c'_2$ |
| $\vdots$ | $\vdots$ |
| $c_n$ | $c'_n$ |

*Pour que le joueur $II$* gagne le jeu $HR_{m,n}(\mathcal{A}, \mathcal{B})$*, il faut et il suffit que les conditions suivantes soient satisfaites :*

1. *$\forall i, j \in \{1, \ldots, k\}$ on a $|C_i| \leq |C_j|$ si, seulement si, $|C'_i| \leq |C'_j|$.*
2. *$(\mathcal{A}, C_1, \ldots, C_k, c_1, \ldots, c_l)$ et $(\mathcal{B}, C'_1, \ldots, C'_k, c'_1, \ldots, c'_l)$ sont isomorphes.*

Ce jeu aussi est complet, dans le sens que si le joueur $II$ n'a pas de stratégie gagnante alors le joueur $I$ en a une. Le théorème suivant prouve l'équivalence entre ce jeu et la logique $MSO(Q_h, Q_r)$.

**THÉORÈME 16** *Le joueur $II$ a une stratégie gagnante pour le jeu $HR_{m,n}(\mathcal{A}, \mathcal{B})$ si, et seulement si, les structures $\mathcal{A}$ et $\mathcal{B}$ satisfont les mêmes énoncés de*

$$MSO(Q_h, Q_r)$$

*ayant, sous forme prénexe, $m$ quantificateurs monadiques suivis de $n$ quantificateurs du premier ordre.*

[1] Les lettres H et R sont pour les initiales de Härtig et Rescher.

**Preuve :** Supposons que la première condition de gain du joueur $II$ n'est pas satisfaite. On a alors :

$$(\mathcal{A}, C_1, \cdots, C_m) \models \bigwedge_{i,j:C_i<C_j} Q_r(X_i, X_j)$$

et

$$(\mathcal{B}, C'_1, \cdots, C'_m) \not\models \bigwedge_{i,j:C_i<C_j} Q_r(X_i, X_j)$$

On quantifie les variables du second ordre en fonction des choix du joueur $I$ au $i^{\text{ème}}$ coup : on quantifie la variable $X_i$ existentiellement s'il choisit $\mathcal{A}$ et universellement sinon, soit $Q_i$ ce quantificateur. Alors la formule

$$Q_1 X_1 \cdots Q_m X_m \bigwedge_{i,j:C_i<C_j} Q_r(X_i, X_j)$$

est satisfaite par $\mathcal{A}$ et pas par $\mathcal{B}$. Si c'est la seconde condition qui échoue, on reprend le jeu $MSO_{m,n}(\mathcal{A}, \mathcal{B})$, et donc il y aura une formule de $MSO$ qui fait notre affaire.

Pour la seconde direction, supposons que le joueur $II$ a une stratégie gagnante pour le jeu $HR_{m,n}(\mathcal{A}, \mathcal{B})$. Les choix des joueurs seront représentés comme dans le tableau suivant :

| $\mathcal{A}$ | $\mathcal{B}$ |
|---|---|
| $C_1$ | $C'_1$ |
| $C_2$ | $C'_2$ |
| $\vdots$ | $\vdots$ |
| $C_m$ | $C'_m$ |
| $c_1$ | $c'_1$ |
| $c_2$ | $c'_2$ |
| $\vdots$ | $\vdots$ |
| $c_n$ | $c'_n$ |

À la fin du jeu, on a les deux structures :

$$(\mathcal{A}, C_1, \ldots, C_k, c_1, \ldots, c_l) \text{ et } (\mathcal{B}, C'_1, \ldots, C'_k, c'_1, \ldots, c'_l)$$

qui satisfont les mêmes formules atomiques, par la condition 2, et la définition de l'isomorphisme, et satisfont les mêmes formules de la forme $Q_h(X_i, X_j)$, pour des variables $X_i, X_j$ interprétées par $C_i, C_j$ dans $\mathcal{A}$ et $C'_i, C'_j$ dans $\mathcal{B}$.

On peut conclure que, pour ce choix, aucune formule ayant comme quantificateurs $\exists$ si le joueur $I$ choisit dans la structure $\mathcal{A}$ et $\forall$ sinon, ne peut séparer les deux

structures. Puisque le jeu autorise le joueur $I$ à choisir, pour tout coup $i$, dans $\mathcal{A}$ ou dans $\mathcal{B}$, et que le joueur $II$ peut toujours contrer, on peut conclure que $\mathcal{A}$ et $\mathcal{B}$ satisfont les mêmes formules de $MSO(Q_r, Q_h)$ sous forme prénexe, ayant $m$ quantificateurs unaires et $n$ quantificateurs individuels. $\square Preuve$

Comme corollaire de ce théorème, on obtient une méthode complète pour prouver si oui ou non un langage est rudimentaire.

**Corollaire 6.1** *Soit $\mathcal{L}$ un langage. Supposons que pour tout $m, n \in \mathbb{N}$, il existe deux structures $\mathcal{A} \in \mathcal{L}$ et $\mathcal{B} \notin \mathcal{L}$ tel que le joueur $II$ a une stratégie gagnante du jeu $HR_{m,n}(\mathcal{A}, \mathcal{B})$. Alors $\mathcal{L}$ n'est pas définissable par un énoncé de $MSO(Q_h, Q_r)$.*

**Remarque.** On peut donner un autre jeu pour cette classe inspiré de la logique $MSO(Maj)$. Ce jeu se déroulera comme le $HR_{m,n}(\mathcal{A}, \mathcal{B})$, mais la condition de gain du joueur $II$ sera :

1. $\forall i \in \{1, \ldots, k\}$ on $Maj(C_i)$ si, et seulement si, $Maj(C'_i)$.
2. $\langle \mathcal{A}, C_1, \ldots, C_k, c_1, \ldots, c_l \rangle$ et $\langle \mathcal{B}, C'_1, \ldots, C'_k, c'_1, \ldots, c'_l \rangle$ sont isomorphes. $\square$

En essayant d'utiliser ces jeux, on se heurte à deux problèmes : le premier est la difficulté de définir des problèmes "simples" qui soient à la frontière de $LinH$ et le second est la difficulté de trouver une stratégie pour les logiques utilisant une alternance de quantificateurs.

Pour restreindre ce jeu de non-définissabilité au fragment existentiel de la logique $MSO(Q_h, Q_r)$, on oblige le joueur $I$ à choisir dans la première structure $\mathcal{A}$ et le joueur $II$ dans la seconde $\mathcal{B}$, voir la sous-section 2.4.3.

## 6.2 Monadic-NP et connexité sur les graphes

Büchi [Büc60] a prouvé ce qui suit : *"Le fragment existentiel de la logique monadique du second ordre sur les structures de mots a le même pouvoir d'expression que la logique monadique du second ordre"*, voir aussi [Str94, Tho97, Pin96]. Par contre, sur les graphes finis, la logique monadique du second ordre et son fragment existentiel ne sont pas égaux. Ce résultat, dû à Fagin [FSV95], a été prouvé en utilisant le prédicat de connexité des graphes finis. Un graphe $G$ est *connexe* si, et seulement si, pour tout $(x, y) \in G$, on peut atteindre $x$ à partir de $y$ par la relation d'accessibilité. On utilisera dans cette section que des graphes non-orientés. Fagin [FSV95] a prouvé que ce prédicat est définissable dans Monadic-coNP mais pas dans Monadic-NP. Michel de Rougemont [MdR87] a prouvé que, même en présence d'une relation binaire de successeur, la connexité des graphes finis reste non-définissable dans $Monadic-NP$.

Les résultats de Fagin et de de Rougemont utilisent des arguments de localité dus à Hanf et Gaifman, voir [EF95]. Pour un exposé sur ces résultats, voir [Fag97, Imm99, Sch96].

Thomas Schwentick [Sch96] a prouvé que, même en présence d'un ordre linéaire sur les sommets du graphe, ceci reste valable. Une des questions qui en découle naturellement est : "Est-ce que la connexité des graphes finis est prouvable dans $\exists MSO$ en présence des prédicats $Bit$ et $<$ qu'Immerman a prouvé équivalent à $(+, \times)$."

On va rappeler le résultat de Schwentick et on va remarquer que la construction qu'il a faite reste valable même en autorisant les quantificateurs de Rescher et de Härtig dans $\exists MSO(<)$. Cette logique est, *a priori*, moins puissante que $\exists MSO(Bit)$.

Soit $n > 0$, un entier. Soit $P = p_1, \cdots, p_l$ une suite de permutations sur $[n] = \{1, \cdots, n\}$. On définit $G_P$, le graphe de $P$, comme suit : Tous les nœuds de $G_P$ sont numérotés $v_{ij}, i = 1, \cdots l+1, j = 1, \cdots, n$, et sont ordonnés lexicographiquement. Les éléments $v_{ij}$ et $v_{i'j'}$ sont reliés par la relation d'accessibilité si, et seulement si :

1. $i' = i + 1$ et $j' = p_i(j)$, ou
2. $i = 1, i' = l + 1$, et $j = j'$.

On peut facilement vérifier que $G_P$ est connexe si, et seulement si,

$$\Pi_{i=1}^{l} p_i$$

est un $n$-cycle.

On définit une fonction de distance par :

$$\delta(v_{ij}, v_{i'j'}) := min(|i - i'|, l - |i - i'|).$$

L'idée de la preuve est la suivante :
- Pour un $n$ assez grand on choisit un graphe $G = G_P$ pour une suite de permutations $P$, chacune contenant toutes les permutation sur $[n]$.
- On prouve que, indépendamment de la façon dont le joueur $I$ choisit les sous-ensembles, pour chaque variable de la formule, il existe une permutation qui peut être remplacée par un nombre de permutations, tel que aucun remplacement n'est détectable par un jeu $FO_k$.
- Finalement, on prouve qu'il existe une combinaison de tels remplacements indétectables dans des sous-suites de $P$ qui donne une suite $P'$ de permutations telles que : Le produit des permutations de $P'$ n'est pas un $n$-cycle, et le graphe, non-connexe, $G' = G_{P'}$ ne peut pas être distingué de $G$ par un $FO_k$ jeu.

Soit $\sigma_1, \cdots, \sigma_{n!}$ une énumération de toutes les permutations sur $[n]$, et soit :

$$\alpha = (\Pi_{i=1}^{n!}\sigma_i)^{-1}.$$

Notons $Q_i$ la suite $2^k$ fois l'identité $\times\sigma_i \times 2^k$ fois l'identité :

$$()() \cdots ()\sigma_i()() \cdots ().$$

Soit $R$ la suite $\alpha, Q_1, \cdots, Q_{n!}$. La suite $P$ sera la suite contenant le $n$-cycle $(12 \cdots n)$ suivi de $n!2^{n!}$ copies de $R$.

Comme le produit de $R$ est l'identité, le produit de toutes les permutations de $P$ est $(1 \cdots n)$. Donc $G_P$ est connexe. On choisira donc $G := G_P$ pour un $n$ assez grand. Notons $K_i$ la $i$-ème copie de $R$ dans le graphe $G$ et $L_{ij}$ le sous-graphe de $G$ représentant $Q_j$ dans le sous-graphe $K_i$.

Soit $G$ coloré arbitrairement avec $c$ couleurs. Le premier but est de prouver que, dans chaque $K_i$, le joueur $II$ peut trouver un ensemble assez grand d'indices $A_i$ tel qu'il a une stratégie gagnante sur chaque paire $L_{ij}, L_{ij'}$, pour $j, j' \in A_i$. Chaque $L_{ij}$ consiste en $q(n) = n(2^k + 2 + 2^k)$ nœuds. Soit $\delta_{ij}(x)$ la distance qui sépare le nœud $x$ des deux colonnes du milieu de $L_{ij}$. Alors, les sous-graphes $L_{ij}, L_{ij'}$ pour $j, j' \in A_i$, diffèrent seulement en leurs permutations centrales, et sont colorés identiquement. Si $j, j' \in A_i$, alors le joueur $II$ a une stratégie gagnante sur $FO_k(L_{ij}, L_{ij'})$, qui respecte $\delta_{ij}$ et $\delta_{ij'}$.

Dans sa preuve, Schwentick [Sch96] combine des modifications pour rendre le graphe $G$ non connexe. Le graphe résultant sera donc $G'$. Remarquons que les sous-ensembles choisis sont les mêmes dans $G$ et $G'$. On a donc une stratégie pour le jeu existentiel $HR_{c,k}(G, G')$. On conclut :

**THÉORÈME 17** *La connexité des graphes finis n'est pas définissable dans*

$$\exists MSO(<, Q_h, Q_r).$$

## 6.3 Le pouvoir d'expression de $\exists MSO(Q_h, Q_r)$ sur les structures de mots

Dans cette section, on va donner une borne supérieure pour les langages définissables dans $\exists MSO(Q_h, Q_r)$ qui est $NTime[n]$, et on donnera ensuite une classe de langages qui est incluse dans $\exists MSO(Q_h, Q_r)$ comme borne inférieure.

**THÉORÈME 18** *Sur les structures de mots, on a :*

$$\exists MSO(Q_h, Q_r) \subseteq NTime[n].$$

**Lemme 6.1** *Toute formule de $\exists MSO(Q_h, Q_r)$ est équivalente à une formule de*

$$\exists MSO(Q_h, Q_r)$$

*où les $Q_h$ et $Q_r$ apparaissent positivement, i.e. dans le champ d'un nombre pair de négations.*

**Preuve :**

On remplace, dans chaque formule, toute occurrence de $\neg Q_h(X, Y)$ par

$$Q_r(X, Y) \vee Q_r(Y, X)$$

et on remplace les occurrences de $\neg Q_r(X, Y)$ par

$$Q_h(X, Y) \vee Q_r(Y, X).$$

Après ce processus on obtient le résultat recherché. $\square Preuve$

**Preuve :**[du théorème.]

On commence par remplacer chaque formule de $\exists MSO(Q_h, Q_r)$ par une formule de $\exists MSO(Q_h, Q_r)$ dans laquelle les $Q_h$ et $Q_r$ apparaissent positivement.

On remplace ensuite les occurrences des formules atomiques $Q_h(Z, T)$ par :

$\exists M \forall xy[(M(x, y) \vee M(y, x)) \rightarrow ((Z(x) \wedge \neg T(x) \wedge T(y) \wedge \neg Z(y)) \vee ((T(x) \wedge \neg Z(x) \wedge Z(y) \wedge \neg T(y)))] \wedge \forall x((Z(x) \wedge \neg T(x)) \rightarrow \exists y(M(x, y) \vee M(y, x))) \wedge \forall x((T(x) \wedge \neg Z(x)) \rightarrow \exists y(M(x, y) \vee M(y, x)))$.

On remplace ensuite les occurrences des formules atomiques $Q_r(Z, T)$ par :

$\exists M \forall xy[(M(x,y) \vee M(y,x)) \rightarrow ((Z(x) \wedge \neg T(x) \wedge T(y) \wedge \neg Z(y)) \vee ((T(x) \wedge \neg Z(x) \wedge Z(y) \wedge \neg T(y)))] \wedge \forall x((Z(x) \wedge \neg T(x)) \rightarrow \exists y(M(x,y) \vee M(y,x))) \wedge \exists x((T(x) \wedge \neg Z(x)) \wedge \forall y \neg(M(x,y) \vee M(y,x)))$.

Ceci est possible car nous pouvons toujours trouver une bijection dont les arcs de son graphe ne se croisent pas (Construction comme pour les parenthèses.). On associe à chaque élément tel que $Z(x) \wedge \neg T(x)$ le plus petit élément tel que $T(y) \wedge \neg Z(y)$ et il y a autant d'éléments satisfaisant $Z$ et $T$ entre les deux et vice versa.

Puisque ce sont des formules closes et que $M$ n'intervient que dans cette formule, on peut mettre les $\exists M$ au début de la formule dans sa forme prénexe. On peut alors mettre la formule sous la forme donnée par Lautemann et *al* [LSS99]. On en conclut que $\exists MSO(Q_h, Q_r) \subseteq NTime[n]$. $\square Preuve$

**THÉORÈME 19** *Sur les structures de mots, on a :*

$$\exists MSO(Q_h, Q_r) \not\subseteq CFL.$$

**Preuve :**

Soit $L = \{a^n b^n c^n | n \in \mathbb{N}\}$ le langage non algébrique sur l'alphabet $\Sigma = \{a, b, c\}$, voir par exemple [Har78].

La formule :

$$\begin{gathered}\exists XYZ \exists xy \forall t[(X(t) \leftrightarrow t \leq x) \wedge (X(t) \leftrightarrow P_a(t)) \wedge \\ (Y(t) \leftrightarrow x < t \leq y) \wedge (Y(t) \leftrightarrow P_b(t)) \wedge \\ (Z(t) \leftrightarrow y < t) \wedge (Z(t) \leftrightarrow P_c(t)) \wedge Q_h(X,Y) \wedge Q_h(Y,Z)].\end{gathered}$$

définit $L$.

On en conclut que $CFL$ ne contient pas $\exists MSO(Q_h, Q_r)$. $\square Preuve$

**Définition 6.2** *La classe des langages algébriques bornés, notée $BCFL$, introduite par Ginsburg en 1966, voir [Pet81] pages 181-182, est la plus petite classe telle que :*

1. *Les langages finis sont dans $BCFL$ ;*
2. *si $L_1$ et $L_2$ sont dans $BCFL$, alors $L_1 L_2$ et $L_1 \cup L_2$ sont dans $BCFL$ ;*
3. *si $L$ est dans $BCFL$, et $u, v$ sont des mots de $\Sigma^*$, alors :*

$$\{u^i L v^i | i \geq 0\},$$

*est dans $BCFL$.*

**Remarque :** La classe $BCFL$ ne contient pas tous les langages réguliers puisque $\Sigma^*$ n'est pas dans $BCFL$ si $\Sigma$ contient au moins deux lettres.

La classe $BCFL$ n'est pas incluse non plus dans la classe des langages réguliers puisque $L = \{a^n b^n | n \in \mathbb{N}\}$ n'est pas régulier et est dans $BCFL$.

**THÉORÈME 20** *Sur les structures de mots, la classe $\exists MSO(Q_h, Q_r)$ contient les langages réguliers et les langages de $BCFL$.*

**Preuve :**

Sur les structures de mots, les langages réguliers correspondent à $\exists MSO$, voir [Büc60].

Prouvons maintenant que $BCFL \subseteq \exists MSO(Q_h, Q_r)$ par induction structurelle des langages de $BCFL$.

**Vérification de la Base.** Soit la formule du premier ordre $\phi_u(i, j)$ qui est satisfaite par un mot $w = w_1 \cdots w_n$ pour des positions $i, j$ si, et seulement si, $u = w_i \cdots w_j$. On en conclut que les langages finis sont définissables par des disjonctions finies de formules du premier ordre.

**Les opérations de clôture.** Pour l'union, il est facile de vérifier que la disjonction de deux énoncés de $\exists MSO(Q_h, Q_r)$ est toujours un énoncé de $\exists MSO(Q_h, Q_r)$.

Pour la concaténation, simple relativisation des variables.

Soit $X$ un ensemble (un prédicat unaire). On définit $S_X(i, j)$, le successeur relatif à $X$ par :

$$i < j \wedge X(i) \wedge X(j) \wedge \forall k(i < k < j \rightarrow \neg X(k)).$$

Soit $Max_X(i)$ le prédicat qui dit que $i$ est le plus grand élément satisfaisant $X$. Soit $Min_X(i)$ le prédicat qui dit que $i$ est le plus petit élément satisfaisant $X$.

Soit $\Phi_L \in \exists MSO(Q_h, Q_r)$ la formule qui caractérise les mots de $L$.

On va marquer les éléments qui forment la dernière lettre de $u$ par un prédicat $X$, et ceux qui forment la première lettre de $v$ par $Y$.

La formule dira qu'il existe $X, Y$ et des positions $x, y$ tels que $Max_X(x)$ et $Min_Y(y)$. on dira que si on a $S_X(i, j)$ alors $\phi_u(i + 1, j)$, et $S_Y(i, j)$ alors $\phi_v(i, j - 1)$.

$$\exists XY \exists xy[(x \leq y) \wedge Max_X(x) \wedge Min_Y(y) \wedge$$
$$\forall ij(S_X(i,j) \rightarrow \phi_u(i+1,j)) \wedge \phi_u(min, Min_X)$$
$$\wedge \forall ij(S_Y(i,j) \rightarrow \phi_v(i,j-1)) \wedge \phi_v(Max_Y, max)]$$

$Q_h(X,Y)$ dira que le mot est de la forme $u^i m v^i$ avec $m \in \Sigma^*$.

Il ne reste plus qu'a relativiser la formule $\Phi_L$ à l'intervale $[x+1, y-1]$. $\square Preuve$

*Question : Si on prouve que les langages de Dyck sont dans $\exists MSO(Q_h, Q_r)$ est-ce-que cela prouve que $\exists MSO(Q_h, Q_r) = NTime[n]$ ?*

**Définition 6.3** *Définissons maintenant la logique $\exists MSO^+(Q_h, Q_r)$. Les formules de cette logique sont les mêmes que celles de $\exists MSO(Q_h, Q_r)$ avec l'autorisation d'avoir des formules atomiques de la forme :*

$$Q_h(\phi(x), \psi(x)) \text{ et } Q_r(\phi(x), \psi(x)),$$

*pour toutes formules du premier ordre $\phi$ et $\psi$, sur la signature augmentée des prédicats monadiques, à une variable libre et des paramètres.*

**Exemple.** La formule :

$$Q_h(0 \leq i \leq x, y \leq i \leq z),$$

définit $z = x + y$. Dans cette formule, $x$, $y$ et $z$ sont des paramètres, ils sont donc libres. $\square Exemple$

**THÉORÈME 21** *Sur les structures de mots, on a :*

$$NTime[n] \subseteq \exists MSO^+(Q_h, Q_r).$$

**Preuve :**

Soit $\mathcal{L} \in NTime[n]$ un langage. D'après le théorème de Lautemann, Schwentick et Schweikhardt [LSS99], il existe une formule :

$$\Phi \equiv \exists M_1 M_2 M_3 \exists \overline{R}(\phi_1 \wedge \phi_2 \wedge \phi_3)$$

dans laquelle les $R_i$ sont unaires, les $M_j$ sont des appariements et les seuls prédicats binaires figurant dans $\phi_i$ sont $M_i$ et $<$. On peut alors traduire cette formule en une formule de $\exists MSO^+(Q_h, Q_r)$ comme suit.

Pour dire qu'il existe un appariement, on dit qu'il y a deux sous-ensembles $O, F$ tels que : $O$ marque les parenthèses ouvrantes ($\exists y M(x,y)$) et $F$ marque les parenthèses

férmantes ($\exists x M(x,y)$). Ceci peut ce faire en disant qu'il y a au moins autant de parenthèses ouvrantes que de parenthèses fermantes dans chaque segment initial du mot, et on obtient à la fin autant de parenthèses ouvrantes que de fermantes :

$$\exists M \equiv \exists OF(\forall x(Q_r(i < x \wedge F(i), i < x \wedge O(i)) \vee Q_h(i < x \wedge F(i), i < x \wedge O(i))) \wedge \forall x \neg(O(x) \wedge F(x)) \wedge Q_h(O(i), F(i))).$$

On note $\exists OF\psi(O,F)$ cet énoncé. La formule atomique $M(x,y)$ est équivalente à la formule $\alpha(x,y)$ :

$$\begin{aligned} &O(x) \wedge F(y) \wedge Q_h(x < i < y \wedge O(i), x < i < y \wedge F(i)) \\ &\wedge \forall t(Q_h(x < i < t \wedge O(i), x < i < t \wedge F(i)) \rightarrow y \leq t) \\ &\wedge \forall t(Q_h(t < i < y \wedge O(i), t < i < y \wedge F(i)) \rightarrow x \geq t). \end{aligned}$$

Notre nouvelle formule $\Phi^*$ sera :

$$\exists O_1 F_1 O_2 F_2 O_3 F_3 \exists \overline{R}(\psi(O_1,F_1) \wedge \phi_1^* \wedge \psi(O_2,F_2) \wedge \phi_2^* \wedge \psi(O_3,F_3) \wedge \phi_3^*),$$

où $\phi_i^*$ correspond à $\phi_i$ dans laquelle on remplace les occurrences de $M(x,y)$ par $\alpha(x,y)$. On remarque facilement que les modèles de $\Phi$ et $\Phi^*$ sont les mêmes.

On conclut que $NTime[n] \subseteq \exists MSO^+(Q_h,Q_r)$. $\square Preuve$

Récapitulons les résultats qu'on a :

$$\exists MSO(Q_h,Q_r) \subseteq NTime[n] \subseteq NLIN \subseteq \exists MSO(+) \subseteq \exists MSO^+(Q_h,Q_r).$$

L'inclusion $MSO(+) \subseteq \exists MSO^+(Q_h,Q_r)$ est une conséquence de l'exemple qu'on a donné au début de la section, et $NLIN \subseteq MSO(+)$ est un résultat de Grandjean et Olive [GO94].

On constate que si $\exists MSO(Q_h,Q_r)$ et $\exists MSO^+(Q_h,Q_r)$ avaient le même pouvoir d'expression, on obtiendrait une nouvelle caractérisation logique de $NTime[n]$ et en même temps de la classe $NLIN$, et on prouverait ainsi leur égalité.

# Chapitre 7

# Conclusion

Dans cette thèse on a exploré un aspect de la théorie des modèles finis qui traite de la caractérisation logique des classes de complexité. Cette discipline est appelée *complexité descriptive*.

Notre première contribution dans ce domaine a été de donner une caractérisation logique de la classe des langages algébriques non ambigus. Cette caractérisation est basée sur les définitions implicites et sur la caractérisation logique des langages algébriques donnéee par Lautemann, Schwentick et Thérien [LST95]. Elle prouve encore une fois, après les résultats de Kolaitis et Lacroix [Kol90, Lac96], que la notion de définition implicite en logique et la notion de calcul non ambigus en théorie de la complexité algorithmique sont reliés. De plus ce résultat relie deux problèmes indécidables. Le premier est celui de savoir si un langage algébrique est, oui ou non, non ambigu, le second est le fait que l'ensemble des formules de la logique $IMP$, voir le chapitre 1 pour la définition, est co-récursivement énumérable complet. On en conclut que, sur les structures de mots, la logique utilisant uniquement la définition implicite de prédicats unaires est décidable, mais qu'en autorisant la définition implicite ne serait-ce que d'un seul prédicat binaire, cette logique devient indécidable.

Une question naturelle se pose : *"Existe-t-il une caractérisation logique des langages algébriques déterministes ?"*

Notre seconde contribution a été de donner une preuve du fait que les langages reconnus par les réseaux de Petri sont reconnaissables en temps non-déterministes linéaire par une machine de Turing. Cette preuve utilise uniquement les caractérisation logiques des langages reconnus par les réseaux de Petri donnée par Parigot et Pelz [PP85] et celle des langages reconnus en temps linéaire par les machine de Turing non-déterministe donnée par Lautemann, Schweikardt et Schwentick [LSS99]. Ceci

va dans la direction des buts de la complexité descriptive qui est de résoudre les problèmes de comparaison de classes de complexité à l'aide de leurs caractérisations logiques. Le résultat le plus célèbre de cette direction est celui d'Immerman et Szelepcsényi, voir par exemple [Imm99], qui a prouvé que la classe des langages reconnus par une machine de Turing non-déterministe en espace logarithmique est clos par complément, *i.e.* $NL = co - NL$.

Enfin on a donné de nouvelles caractérisations logique de la classe des langages rudimentaires. Ces caractérisations sont obtenues en comparant le pouvoir d'expression des logiques à la logique de Lynch et Immerman [Lyn82, Imm87] caractérisant les langages rudimentaires. Ces caractérisations ont deux intérêts. Le premier est de donner une définition des langages rudimentaires sans opérations arithmétiques. La seconde est l'introduction d'un jeu à la Ehrenfeucht-Fraïssé qui est susceptible de prouver si, oui ou non, un langage est rudimentaire. À ma connaissance, c'est la première méthode pour prouver qu'un langage n'est pas rudimentaire. Cette méthode serait plus intéressante si elle avait permis de trouver une relation qui ne soit pas rudimentaire. Mais, je vais reprendre la phrase de Esbelin et More [EM98] :

> *"However, it is difficult to exhibit a natural arithmetical relation which can be proved not to be rudimentary."*

On se heurte à un autre problème qui a été énoncé par Fagin à Luminy en juin 2000[1] qui demande une stratégie gagnante pour un jeu avec plus d'une phase de coloration, *i. e.* choix des sous-ensembles.

Vu ces difficultés, on s'est alors interessé à la partie existentielle de cette logique. En regardant attentivement la preuve de Schwentick [Sch96], du fait que la connexité des graphes finis n'est pas définissable dans la logique existentielle monadique du second ordre avec ordre, on remarque que sa construction fournit aussi une preuve du fait que la connexité des graphes finis n'est pas définissable dans la logique existentielle monadique du second ordre munie de l'ordre et authorsant l'utilisation des quantificateurs de Rescher et Härtig sur les variables unaires.

Ce résultat est une réponse partielle à la question plus importante : *"Est-ce-que la connexité des graphes finis est définissable dans la logique existentielle monadique du second ordre avec addition ?"* On peut maintenant attaquer cette question en comparant le pouvoir d'expression de $\exists MSO(+)$ et celui de $\exists MSO(Q_h, Q_r)$.

[1] Voir les problèmes ouverts sur le site `http ://www-mgi.informatik.rwth-aachen.de/FMT/`

# Bibliographie


[Ajt83] M. Ajtai, $\Sigma_1^1$-*Formulae on Finite Structures,* Annals of Pure and Applied Logic, 24, 1983, 1-48.

[AB84] M. Ajtai & M. Ben-Or, *A Theorem on Probabilistic Constant Depth Computations,* Proc. 16th ACM STOC, 1984, 471-475.

[AFS00] M. Ajtai, R. Fagin & L. Stockmeyer, *The Closure of Monadic NP,* J. Computer and System Sciences, 60, 2000, 660-716.

[ABB97] J. M. Autebert, J. Berstel and L. Boasson, *Context Free Languages and Pushdown Automata*, in Handbook of Formal Languages, vol 1, Springer Verlag, 1997, 111-174.

[BIS90] D. M. Barrington, N. Immerman & H. Straubing, *On Uniformity Within* $NC^1$, Journal of Computer and System Science, 41, 1990, 274-306.

[Ben61] J. H. Bennett, *On Spectra,* Doctoral dissertation, Princeton University, 1962.

[BG70] R. Book and S. Greibach, *Quasi Realtime Languages*, Math. System Theory, 4, 1970, 97-111.

[Büc60] J. R. Büchi, *Weak Second-Order Arithmetic and Finite Automata*, Zeit.Math.Logik und Grund. Math., 6, 1960, 66-92.

[CK90] C. C. Chang & H. J. Keisler, *Model Theory,* third edition, North Holland, 1990.

[CL93] R. Cori & D. Lascar, *Logique mathématique,* Volume I et II, Masson, 1993.

[Cou90] B. Courcelle, *The Monadic Second-Order Logic of Graphs I : Recognizable sets of finite graphs*, Information and Computation, 85, 1990, 12-75.

[DDLW98] A. Dawar, K. Doets, S. Lindell & S. Weinstein, *Elementary Properties of the Finite Ranks,* in Mathematical Logic Quarterly, 1998.

[DGS86] L. Denenberg, Y. Gurevich & S. Shelah, *Definability by Constant Depth Polynomial-Size Circuits,* Information and Control, 70, 1986, 216-240.

[MdR87] M. de Rougement, *Second-order and inductive definability on finite structures,* Zeit. für Math. Logik und Grund. der Math., 33, 1987, 47-63.

[Don70] J. E. Doner, *Tree Acceptors and Some of their Applications*, Journal of Computer and System Science, 4, 1970, 406-451.

[DLM98] A. Durand, C. Lautemann & M. More, *Counting Results in Weak Formalisms,* Rapport SDAD, Université de Caen, 1998.

[EF95] H. D. Ebbinghaus & J. Flum, *Finite Model Theory*, Springer, 1995.

[EGG98] T. Eiter, G. Gottlob & Y. Gurevich, *Existential Second order Logic Over Strings*, in Proc. of the $13^{th}$ Symp. of Logic In Computer Science, 1998, 16-27.

[Elg61] C. C. Elgot, *Decision Problems of Finite Automata and related Arithmetics,* Trans. of the AMS, 98, 1961, 21-52.

[EM98] H.-A. Esbelin & M. More, *Rudimentary relations and primitive recursion : A toolbox,* Theoretical Computer Science, 193, 1998, 129-148.

[Fag74] R. Fagin, *Generalized First-order Spectra and Polynomial-time Recognizable Sets,* R. Karp editor, The Complexity of Computation, SIAM-AMS Proceedings, vol 7, 1974, 27-41.

[Fag97] R. Fagin, *Easier Ways to Win Logical Games,* in Descriptive Complexity and Finite Models, N. Immerman and P. Kolaitis editors, 1997, 1-32.

[FSV95] R. Fagin, L. Stockmeyer & M. Vardi, *On monadic NP vs. Monadic co-NP*, Information and Computation, 120, 1995, 78-92.

[FSS84] M. Furst, J. B. Saxe & M. Sipser, *Parity, Circuits and the Polynomial Time Hierarchy,* Mathematical System Theory, 17, 1984, 13-27.

[GS84] F. Gécseg & M. Steinby, *Tree Automata,*Akademiai Kiado, Budapest, 1984.

[Géc01] F. Gécseg, Communication personnelle 2001.

[Got01] Jorg Gottlob, Communication personnelle, Rapport de thèse 2001.

[Grä92] E. Grädel, *Capturing Complexity Classes by Fragments of Second-Order Logic,* Theor. Comp. Science, 101, 1992, 35-57.

[GR99] E. Grädel & E. Rosen, *Two Variables Description of Regularity,* in Proc. of the $14^{th}$ Symp. of Logic in Computer Science, 1999.

[GO94] E. Grandjean & F. Olive, *Monadic Logical Definability of NP-Complete Problems*, in CSL'94, LNCS 933, 1995, 190-204.

[Gur84] Y. Gurevich, *Towards Logic Tailored for Computational Complexity,* Computation and Proof Theory, Lecture Notes in Mathematics 1104, 1984, 175-216.

[Hac01] Y. Hachaïchi, *Bounded Arithmetics and Descriptive Complexity,* manuscrit, accessible par *e-mail* à l'auteur :
`hachai@logique.jussieu.fr`

[HP93] P. Hajek & P. Pudlak, *Metamatematics of First-Order Arithmetic,* Springer-Verlag, 1993.

[Har78] M. A. Harrison, *Introduction to Formal Language Theory*, Addison Wesley, 1978.

[Här65] H. Härtig, *Über einen Quantificator mit zwei Wirkungsbereichen*, in Kalmar ed., Colloquium on Foundations of Mathematics, Mathematical Machines and Their Applications, 1965, 31-36.

[Hot78] G. Hotz, *Normal form transformations of context-free languages*, Acta Cybernetica, 4, 1978, 65-84.

[Imm87] N. Immerman, *Languages That Capture Complexity Classes,* SIAM Journal of Computing, 16, 1987, 760-778.

[Imm99] N. Immerman, *Descriptive Complexity*, Graduate Texts in Computer Science, Springer, New York, 1999.

[KH82] C. Kent & B. Hodgson, *An Arithmetical Characterization of $NP$,* Theoretical Computer Science, 21, 1982, 255-267.

[KH83] C. Kent & C. Hodgson, *A Normal Form for Arithmetical Characterization of $NP$-sets,* Journal of Computer and System Science, 27, 1983, 378-388.

[Kol90] P. Kolaitis, *Implicit definability on finite structures and unambiguous computations*, Proc. 5th Symp. of Logic In Computer Science, 1990, 168-180.

[Lac96] Z. Lacroix, *Bases de Données : des Relations Implicites aux Relations Contraintes,* Thèse de doctorat, Université de Paris-Sud, 1996.

[LdR93] R. Lassaigne & M. de Rougemont, *Logique et fondements de l'informatique,* Hermès, 1993.

[LdR96] R. Lassaigne & M. de Rougemont, *Logique et Complexité,* Hermès, 1996.

[LMSV98] C. Lautemann, P. McKenzie, T. Schwentick & H. Vollmer, *The Descriptive Complexity Approach to $LOGCFL$,* Electronic Colloquium on Computational Complexity, Report No. 59, 1998.

[LSS99] C. Lautemann, N. Schweikardt and T. Schwentick, *A Logical Characterization of Nondeterministic Linear Time on Turing Machines,* STACS '99, Lecture Notes in Computer Science 1563, 143-152 .

[LST95] C. Lautemann, T. Schwentick and D. Thérien, *Logics for Context Free Languages*, CSL'95, Lecture Notes in Computer Science, Volume 933, 205-216.

[Lyn82] J. Lynch, *Complexity Classes and Theories of Finite Models,* Math. System Theory, 15, 1982, 127-144.

[Mat98] O. Matz, *Dot-Depth and Monadic Quantifier Alternation Over Pictures*, Habilitation, RWTH Aachen, juin 1999.

[MP71] R. McNaughton & S. Papert, *Counter Free Automata*, M.I.T. Press, 1971.

[MW67] J. Mezei & J. B. Wright, *Algebraic Automata and Context Free Sets.*, Information and Control, 1967, 3-29.

[MO97] M. More & F. Olive, *Rudimentary Languages and Second-Order Logic,* Mathemtical Logic Quaterly, 43, 1997, 419-426.

[Myh60] G. Myhill, *Linear bounded automata,* Tech. note 60-165, Wright-Patterson Air Force Base, OH, 1960.

[Ott97] M. Otto, *Bounded Variable Logics and Counting,* Lecture Notes in Logic, volume 9, Springer Verlag, 1997.

[Pap94] C. Papadimitriou, *Computational Complexity,* Addison Wesley, 1994.

[PP85] M. Parigot & E. Pelz, *A Logical Approch of Petri Net Languages*, Theoretical Computer Science, 39, 1985, 155-169.

[PW86] J. Paris & A. Wilkie, *Counting in $\Delta_0$ Sets,* Fundamenta Mathematicae, 127, 1986, 67-76.

[Pet81] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice Hall, 1981.

[Pin96] J.-E. Pin, *Logic, Semigroups and Automata on Words*, Annals of Mathematics and Artificial Intelligence, 16, 1996, 343-384.

[Qui46] W. Quine, *Concatenation as a basis for arithmetic from a logical point of view,* Journal of Symbolic Logic, 11, 1946, 105-114.

[Rab69] M. O. Rabin, *Decidability of Second-order Theories and Automata on infinite trees,* Trans. AMS, 141, 1969, 1-35.

[Res62] N. Rescher, *Plurality Quantification,* Journal of Symbolic Logic, 27, 1962, 373-374.

[Ruh99] M. Ruhl, *Addition and Counting Cannot Express Deterministic Transitive Closure,* in Proceeding of *IEEE* Logic in Computer Science, 1999, 326-334.

[Sch96] T. Schwentick, *On winning Ehrenfeucht games and monadic NP,* Annals of Pure and Applied Logic, 79, 1996, 61-92.

[Smu61] R. Smullian, *Theory of Formal Systems,* Annals of Mathematical Studies, 47, Princeton University Press, 1961.

[Sto76] L. Stockmeyer, *The Polynomial Time Hierarchy*, Theoretical Computer Science, 3, 1977, 1-22.

[Str94] H. Straubing, *Finite Automata, Formal Logic, and Circuit Complexity*, Birkhäuser, 1994.

[TW68] J. W. Thatcher & J. B. Wright, *Generalized Finite Automata With Application to a Decision Problem of Second Order Logic*, Mathematical System Theory, 2, 1968, 57-82.

[Tho97] W. Thomas, *Languages, Automata and Logic*, in Handbook of formal languages, vol 3, Springer Verlag, 1997, 389-455.

[Tra61] B. A. Trakhtenbrot, *Finite Automata and the Logic of Monadic Second Order Predicates*, Dokl. Akad. Nauk SSSR, 140, in russian, 1961, 326-329.

[Vää97] J. Väänänen, *Generalized Quantifiers,* in Bulletin of the EATCS, 62, 1997, 115-136.

[Woo81] A. R. Woods, *Some problems in logic and number theory, and their connection*, Doctoral Dissertation, University of Manchester, 1981.

[Wra78] C. Wrathall, *Rudimentary Predicates and Relative Computation*, SIAM Journal of Comp., 7, 1978, 194-209.